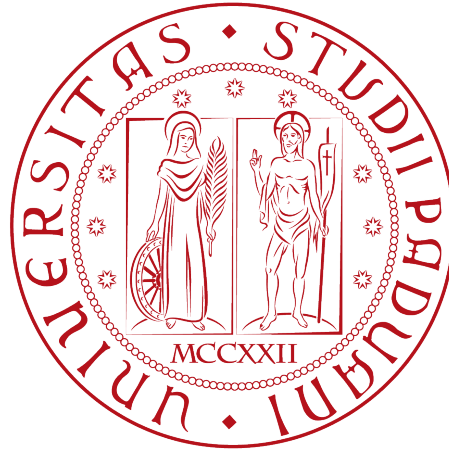


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**ADeQA: una *Progressive Web App* per il  
controllo qualità manuale in filiere produttive  
industriali**

*Tesi di laurea*

*Relatore*

Prof. Vardanega Tullio

*Laureando*

Cusin Matteo

---

ANNO ACCADEMICO 2022/2023



Dedicato a Giulia Cecchettin, coetanea a cui la vita è stata ingiustamente sottratta  
poco prima della conclusione degli studi universitari.  
Che la terra le sia lieve.

# Sommario

L'elaborato descrive i processi, gli strumenti e le metodologie coinvolte nello sviluppo di una *Progressive Web App*<sup>1</sup>, ovvero di un'applicazione *web* sviluppata per fornire un'esperienza simile a quella offerta da un'applicazione nativa, atta all'inserimento manuale di dati relativi al controllo qualità<sup>2</sup> di filiere produttive<sup>3</sup> industriali.

Nel dominio applicativo di interesse dell'elaborato:

- **Controllo qualità:** è un processo atto a garantire che i prodotti / i servizi richiesti soddisfino degli standard prefissati;
- **Filiera produttiva:** è la sequenza delle lavorazioni, effettuate in successione, aventi come fine la trasformazione delle materie prime in un prodotto finito (ingl. *supply chain*).

Il prodotto software, sviluppato nel corso del tirocinio presso l'azienda **Trizeta S.r.l** (d'ora in avanti **Trizeta**) ha la peculiarità di doversi integrare in un software già presente nella suite aziendale e, al tempo stesso, essere in grado di eseguire in maniera del tutto indipendente replicando, all'occorrenza, alcune delle funzionalità presenti in esso.

## Struttura del testo

Il corpo principale della relazione è suddiviso in 4 capitoli:

**Il primo capitolo** descrive il contesto in cui sono state svolte le attività di tirocinio curricolare, concludendo con una riflessione relativa al rapporto tra l'azienda ospitante e l'innovazione all'interno di processi e strumenti aziendali;

**Il secondo capitolo** approfondisce le motivazioni che hanno consentito l'unione delle volontà del proponente e del sottoscritto al fine di acquisire nuove conoscenze e competenze (per il sottoscritto) e risolvere determinati bisogni relativi al dominio aziendale (per **Trizeta**);

**Il terzo capitolo** descrive i processi, gli strumenti e le modalità di esecuzione delle attività lavorative, oltre ai risultati conseguiti;

**Il quarto capitolo** esegue una retrospettiva sul progetto, mettendo in relazione le competenze acquisite durante il percorso didattico e le competenze richieste dal tirocinio curricolare.

---

<sup>1</sup>*Progressive Web App*

<sup>2</sup>Controllo qualità

<sup>3</sup>Filiera produttiva

Di seguito all'ultimo capitolo, si trovano le sezioni:

- **Appendice A:** questa sezione riporta una breve analisi relativamente ai modelli di ciclo di vita del *software* detti "*agili*";
- **Acronimi e abbreviazioni:** ogni voce di questa sezione contiene un collegamento al relativo termine nel **Glossario**;
- **Glossario:** si riportano le definizioni dei termini specifici di dominio, collegandoli (se presenti) ai relativi acronimi o abbreviazioni; dopo ogni definizione, vengono resi disponibili dei collegamenti alle pagine in cui essi sono utilizzati;
- **Bibliografia:** in questa sezione vengono riportate le fonti di informazione utilizzate per dare definizione ad un concetto, indicando eventuali collegamenti ipertestuali esterni, porzione di testo in cui sono state citate e termine al quale si riferiscono.

## Convenzioni tipografiche

Riguardo la stesura del testo, sono state adottate le seguenti convenzioni tipografiche:

- Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel capitolo **Glossario**;
  - Al primo utilizzo di uno dei termini precedentemente indicati, verrà fornita un'essenziale definizione in sede di utilizzo;
  - Il solo primo utilizzo di uno dei termini di cui sopra sarà accompagnato da una nota a piè di pagina contenente il riferimento al termine nel capitolo **Glossario**.
- I termini in lingua straniera, nomi propri ed i termini facenti parte del gergo tecnico, sono evidenziati usando lo stile *corsivo*;
- Il nome dell'azienda ospitante del periodo di tirocinio, i nomi dei capitoli del documento ed i termini chiave delle attività di tirocinio sono evidenziati usando lo stile **grassetto**;
- Le parole rilevanti in sezioni molto ampie di testo saranno evidenziate usando lo stile **grassetto**;
- I termini che definiscono da soli le voci di un elenco (sia esso numerato o puntato) saranno evidenziati usando lo stile **grassetto**;
- Ogni voce di un elenco (puntato e numerato) sarà terminato con un punto e virgola ad eccezione dell'ultima voce, che terminerà con un punto;
- Ogni collegamento a pagine *web* sarà di colore rosso e font **monospaziato**;
- Elementi di codice sorgente avranno font **monospaziato**;
- Le fonti delle immagini saranno riportate come nota a piè di pagina, indicando graficamente la pagina principale di appartenenza.

# Ringraziamenti

*Innanzitutto, desidero esprimere la mia gratitudine al Prof. Vardanega Tullio, tutor del tirocinio curricolare e relatore della mia tesi di laurea, per il continuo sostegno e la disponibilità dimostrati a partire dal primo semestre del terzo anno, durante il corso "Ingegneria del Software" da Egli presieduto.*

*Ringrazio di cuore tutto il team Trizeta per avermi fatto sentire parte integrante del gruppo ed avermi dato la possibilità di eseguire le attività in un ambiente sereno e stimolante.*

*Desidero ringraziare con affetto chi mi è stato vicino durante gli anni di studio, in particolar modo mia sorella Sabrina ed i miei zii Marina e Rossano per il costante sostegno e la fiducia riposta in me.*

*Ho desiderio di ringraziare i miei amici, in particolare Annalisa e Nicola, per i momenti di crescita personale e professionale condivisi ed i legami maturati.*

*Padova, Dicembre 2023*

Cusin Matteo

# Indice

<b>1</b>	<b>Contesto di svolgimento delle attività</b>	<b>1</b>
1.1	Introduzione all'azienda ospitante . . . . .	1
1.2	Prodotti e servizi . . . . .	2
1.3	Processi interni . . . . .	4
1.4	Rapporto con l'innovazione . . . . .	6
<b>2</b>	<b>Motivazioni alla base del tirocinio</b>	<b>7</b>
2.1	Strategia aziendale . . . . .	7
2.2	Problematiche poste in essere . . . . .	9
2.3	Obiettivi . . . . .	10
2.4	Vincoli . . . . .	11
2.5	Pianificazione . . . . .	11
2.6	Scelta del tirocinio . . . . .	13
<b>3</b>	<b>Elementi caratterizzanti del progetto</b>	<b>15</b>
3.1	Stile lavorativo . . . . .	15
3.2	Strumenti utilizzati . . . . .	16
3.2.1	Strumenti di sviluppo . . . . .	16
3.2.2	Strumenti di versionamento . . . . .	20
3.2.3	Strumenti di documentazione . . . . .	20
3.3	Analisi dei requisiti . . . . .	21
3.3.1	Casi d'uso . . . . .	21
3.3.2	Requisiti . . . . .	23
3.4	Progettazione . . . . .	26
3.4.1	Internazionalizzazione e localizzazione . . . . .	26
3.4.2	Interfaccia grafica . . . . .	28
3.4.3	Architettura . . . . .	32
3.5	Codifica . . . . .	34
3.6	Verifica . . . . .	38
3.7	Validazione . . . . .	39
3.8	Risultato finale . . . . .	39
3.8.1	Statistiche qualitative e quantitative . . . . .	40
3.8.2	Interfaccia grafica . . . . .	41
<b>4</b>	<b>Contesto di svolgimento delle attività</b>	<b>44</b>
4.1	Soddisfacimento degli obiettivi prefissati . . . . .	44
4.2	Competenze e conoscenze acquisite . . . . .	44

<i>INDICE</i>	vii
4.3 Competenze curriculari e lavorative . . . . .	44
<b>A Metodologie agili</b>	<b>45</b>
<b>Acronimi e abbreviazioni</b>	<b>46</b>
<b>Glossario</b>	<b>47</b>
<b>Bibliografia</b>	<b>51</b>



# Elenco delle figure

1.1	Interfaccia di un gestionale <i>Trizeta</i> . . . . .	1
1.2	Funzionalità di un <i>software WMS</i> . . . . .	2
1.3	Funzionalità di un <i>software DAM</i> . . . . .	3
1.4	Funzionalità di un <i>software MES</i> . . . . .	3
1.5	Attività di sviluppo durante un periodo agile . . . . .	5
1.6	Componente grafico <i>TreeList</i> di <i>DevExtreme</i> . . . . .	6
2.1	Interfaccia del <i>software ADeMES</i> . . . . .	8
2.2	Interfaccia del prodotto <i>ADeQA</i> . . . . .	9
2.3	Schema sintesi del concetto di <i>black-box</i> . . . . .	9
2.4	Diagramma <i>Gantt</i> di suddivisione delle attività . . . . .	13
2.5	Interfaccia del <i>software Figma</i> . . . . .	14
3.1	Confronto tra la progettazione di interfacce e di esperienze utente . .	16
3.2	Diagramma del <i>runtime Node.js</i> e dei pacchetti in esso installati . . .	18
3.3	Diagramma dei <i>framework</i> , delle librerie e dei linguaggi usati . . . .	19
3.4	Diagramma degli <i>editor</i> di testo usati . . . . .	19
3.5	Diagramma degli strumenti di documentazione . . . . .	20
3.6	Diagramma dei casi d'uso UCF-2 e UCE-3. . . . .	22
3.7	Diagramma dei casi d'uso UCF-5 e UCE-5. . . . .	23
3.8	Interfaccia ideata per la prima schermata di autenticazione . . . . .	28
3.9	Interfaccia ideata per la seconda schermata di autenticazione . . . .	29
3.10	Interfaccia ideata per la visualizzazione principale . . . . .	30
3.11	Visualizzazione principale adattata a dispositivi mobili . . . . .	31
3.12	Visualizzazione principale con colori chiari . . . . .	31
3.13	Diagramma delle classi del servizio di autenticazione . . . . .	32
3.14	Diagramma delle classi del componente di gestione delle fasi . . . . .	33
3.15	Diagramma delle classi del componente di modifica dei dati di qualità	33
3.16	Diagramma delle classi del componente di visualizzazione dei dati di qualità . . . . .	34
3.17	Modello di sviluppo del prodotto adottato . . . . .	39
3.18	<i>Code coverage</i> di progetto . . . . .	40
3.19	Schermata di autenticazione - passo 1 . . . . .	41
3.20	Schermata di autenticazione - passo 2 . . . . .	42
3.21	Schermata principale . . . . .	42
3.22	Intestazione - visualizzazione <i>desktop</i> . . . . .	43
3.23	Barra laterale . . . . .	43

3.24	Intestazione - visualizzazione <i>mobile</i> . . . . .	43
------	--	----

## Elenco delle tabelle

2.1	Distribuzione delle ore di lavoro . . . . .	13
3.1	Requisiti funzionali . . . . .	24
3.2	Requisiti di qualità . . . . .	25
3.3	Requisiti prestazionali . . . . .	25
3.4	Requisiti di vincolo . . . . .	25
3.5	Tabella riassuntiva delle classificazioni dei requisiti . . . . .	25
3.6	Tabella riassuntiva dei requisiti . . . . .	26
3.7	Tabella riassuntiva dei requisiti soddisfatti . . . . .	40
3.8	Statistiche quantitative sul prodotto . . . . .	41

# Capitolo 1

## Contesto di svolgimento delle attività

Questo capitolo si occupa di fornire informazioni in merito a **Trizeta**, azienda ospitante del tirocinio, al settore in cui essa opera (quindi anche ai beni e servizi offerti), al suo rapporto con l'introduzione di novità / con il miglioramento di processi e strumenti già in uso ed ai processi in essa utilizzati.

Le informazioni riportate di seguito sono frutto di osservazioni personali, dialoghi avuti nel corso del tirocinio e ricerche svolte in totale autonomia.

### 1.1 Introduzione all'azienda ospitante

**Trizeta** è una *software house*<sup>4</sup>, ovvero un'azienda che si occupa dello sviluppo e della commercializzazione di *software*, specializzata nella consulenza e nello sviluppo di prodotti per aziende che desiderano l'automazione (totale o parziale) delle proprie attività industriali (compresa la gestione del magazzino); essa consente inoltre alle aziende clienti di gestire le proprie risorse digitali multimediali (i cosiddetti *digital assets*<sup>5</sup>).

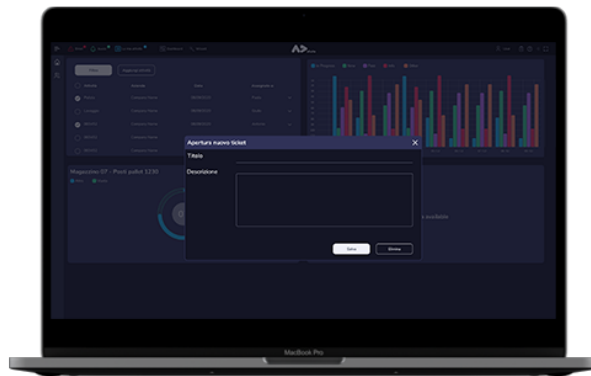


Figura 1.1: Interfaccia di un gestionale **Trizeta**<sup>6</sup>

---

<sup>4</sup> *Software house*

<sup>5</sup> *Digital asset*

<sup>6</sup> Fonte: <https://trizeta.com>

L'azienda è ubicata a *Monselice (Padova)* e dispone all'incirca di una decina di dipendenti *IT*<sup>7</sup> (informatici) tra loro eterogenei per anni di esperienza nel settore informatico, età anagrafica, e *stack* tecnologico<sup>8</sup> abitualmente utilizzato (tecnologie utilizzate e ambito di utilizzo delle stesse).

Recentemente **Trizeta** è entrata a far parte di *SYS-DAT Group*: è un gruppo di aziende specializzate nello sviluppo e manutenzione di prodotti *software* rivolti ad aziende appartenenti a vari settori quali il settore moda (settore di origine di *SYS-DAT*, azienda fondatrice del gruppo) ed il settore alimentare.

## 1.2 Prodotti e servizi

Come già indicato nella sezione precedente, **Trizeta** intrattiene relazioni commerciali esclusivamente di tipo *B2B*<sup>9</sup>: questa visione si riflette inevitabilmente sui prodotti offerti e sull'insieme dei requisiti utente soddisfatti dai prodotti commercializzati.

Di seguito, un breve elenco di software che ho potuto visionare personalmente e, relativamente all'ultima voce in lista, studiare ai fini di comprendere meglio le finalità dello *stage* e la visione dell'azienda:

- *ADeWMS*: è un *WMS*<sup>10</sup> (gestionale relativo al contenuto e alle attività di magazzino) in grado di integrarsi con software *ERP*<sup>11</sup> (gestionale per tutti i processi aziendali) e gestire ordini commerciali, consegne e relativa documentazione;



Figura 1.2: Attività gestite da un software *WMS*<sup>12</sup>

<sup>7</sup> *Information Technology (IT)*

<sup>8</sup> *Stack tecnologico*

<sup>9</sup> *Business to business*

<sup>10</sup> *Warehouse management system*

<sup>11</sup> *Enterprise resource planning*

<sup>12</sup> Fonte: <https://www.logisticaefficiente.it>

- *P4NDOR4*: è un *DAM*<sup>13</sup> (gestionale per *digital assets*, risorse digitali, aziendali) con possibilità di richiedere delle risorse direttamente a *Trizeta*;



Figura 1.3: Funzionalità di un *software DAM*<sup>14</sup>

- *ADeMES*: è un *MES*<sup>15</sup> (*software* di gestione delle attività produttive aziendali) di particolare interesse in quanto direttamente coinvolto ai fini del tirocinio.



Figura 1.4: Funzionalità di un *software MES*<sup>16</sup>

<sup>13</sup> *Digital asset management*

<sup>14</sup>Fonte: <https://vitolavecchia.altervista.org/>

<sup>15</sup> *Manufacturing execution system*

<sup>16</sup>Fonte: <https://www.systema.com>

*ADeMES* è dotato delle seguenti caratteristiche (rilevanti ai fini del tirocinio):

- Presenza di un'area personale per ogni operatore (lavoratore in linea di produzione);
- Lista di "fasi" di lavorazione attive (schedulate o in esecuzione);
- Possibilità di allegare documenti e note testuali ad ogni fase di lavorazione attiva; queste funzionalità hanno molteplici scopi e, tra questi, vi era anche la registrazione di dati di qualità prima dell'inizio del periodo di tirocinio.

### 1.3 Processi interni

Il *team* aziendale utilizza un modello di ciclo di vita del software detto *agile*, ovvero ha una visione orientata all'ottimizzazione del flusso di lavoro (evitando tempi morti e uso di risorse senza ottenere valore in cambio), e consentire una risposta rapida alle variazioni delle esigenze del cliente anche in stadi avanzati dello sviluppo <sup>17</sup>.

- **Gestione di progetto:** in relazione alla gestione di progetto, ho potuto assistere (direttamente o indirettamente) alle seguenti attività:
  - **Definizione degli obiettivi e delle risorse:** la definizione degli obiettivi e delle risorse di progetto avviene dopo dialogo diretto con le industrie clienti coinvolte nel progetto: in questa occasione si cerca di analizzare a fondo i risultati desiderati e le modalità di raggiungimento degli stessi;
  - **Pianificazione:** la pianificazione delle attività avviene a partire dalla definizione degli obiettivi e delle risorse di progetto (previo dialogo, come precedentemente indicato) basandosi su esperienze pregresse e sulla disponibilità di capitale umano e risorse economiche;
  - **Comunicazione con gli *stakeholders***<sup>18</sup>: la comunicazione con i "portatori d'interesse" (coloro i quali hanno interesse nella buona riuscita del progetto) avviene tramite colloquio (preferibilmente in presenza, *online* in caso di necessità) ed è fondamentale per dare prova di avanzamento tangibile nei modi e tempi indicati o, in caso contrario, motivare eventuali discrepanze tra la pianificazione e la realtà.
- **Sviluppo:** le attività qui descritte sono svolte ciclicamente in periodi di tempo più o meno ampi, la cui ampiezza deriva dagli obiettivi prefissati per il singolo periodo.
  - **Analisi dei requisiti:** l'attività di analisi dei requisiti, come da metodologia agile, viene eseguita dopo aver dialogato con il cliente (inizialmente in una sede del cliente, data la frequente presenza di vincoli *hardware* dovuti al settore in cui *Trizeta* opera); i bisogni espressi vengono modellati in *user-stories* (brevi descrizioni di una caratteristica / funzionalità del software, scritta dal punto di vista dell'utente finale o del cliente) e successivamente raffinati fino all'ottenimento di requisiti utente;

---

<sup>17</sup> [Appendice: metodologie agili](#)

<sup>18</sup> [Stakeholder](#)

- **Progettazione:** l'attività di progettazione si basa sui requisiti utente del punto precedente ed ha il fine di ideare la struttura del *software* richiesto perseguendo certi scopi (quali possono essere la mantenibilità del codice e l'usabilità);
- **Codifica:** l'attività di codifica è conseguenza della progettazione e serve per creare il prodotto come da specifica;
- **Verifica:** l'attività di verifica serve per constatare che effettivamente il prodotto esegue come da progettazione; questa attività si basa su test automatizzati e manuali (la seconda tipologia è usata in particolare per verificare che determinate funzionalità grafiche dei prodotti siano funzionanti su dispositivi diversi, dato che molti prodotti *Trizeta* devono funzionare su *tablet* e garantire l'usabilità su *smartphone*);
- **Validazione:** l'attività di validazione si concretizza con un incontro assieme ad alcuni rappresentanti dell'azienda cliente: essi stabiliscono se l'avanzamento prodotto durante il periodo di attività ha soddisfatto (quindi vi è accettazione dell'avanzamento proposto) o meno (vi è rifiuto totale/parziale delle modifiche effettuate al prodotto) le aspettative e gli accordi presi all'inizio.

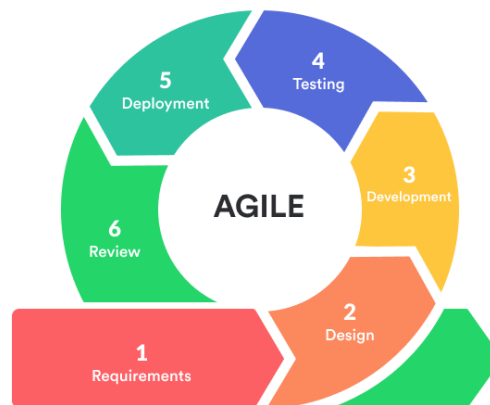


Figura 1.5: Attività di sviluppo durante un periodo agile <sup>19</sup>

#### • Manutenzione

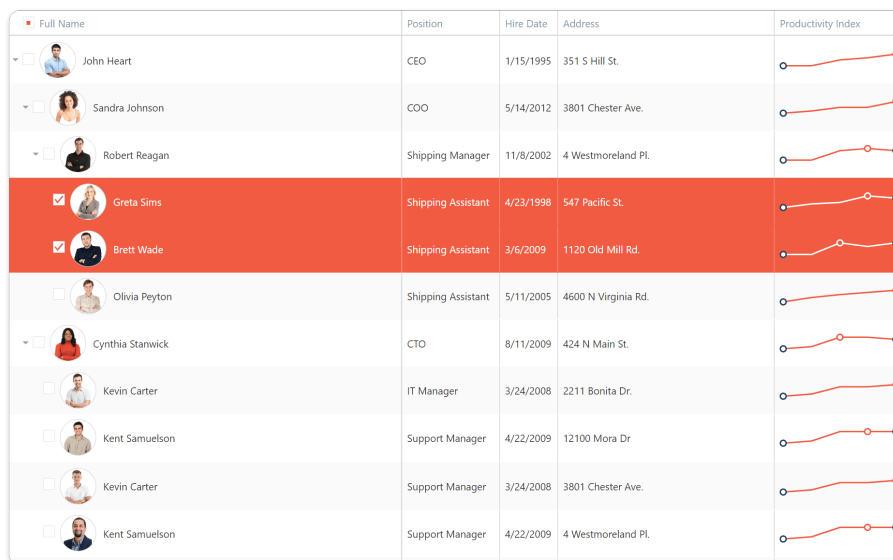
- Risoluzione di problemi: l'azienda ospitante si avvale di un sistema di *ticketing* per la gestione delle segnalazioni (disservizi o anomalie) e comunicazione della loro risoluzione, limitando al minimo i contatti sincroni (non sempre possibili in tempi utili alla risoluzione del problema);
- Cambiamenti evolutivi: in *Trizeta* i cambiamenti evolutivi possono avere origine da un'esigenza espressa da un cliente, da un'idea di un dipendente o dall'aggiornamento di librerie e *framework* utilizzati; la valutazione di un cambiamento è (per quanto ho potuto osservare) un momento in cui ognuno può esprimere un parere motivato.

<sup>19</sup>Fonte: <https://mlsdev.com>

## 1.4 Rapporto con l'innovazione

*Trizeta* ha manifestato a più riprese il suo rapporto con l'innovazione<sup>20</sup> (ovvero l'introduzione di novità e il miglioramento di processi e tecnologie impiegati) nel corso del rapporto lavorativo intercorso:

- *ADeGO*: questo *software* è stato sviluppato per consentire servizi di assistenza da remoto tramite **realtà aumentata**<sup>21</sup> (una tecnica per aggiungere informazioni alla realtà circostante all'utente, utilizzando adeguati supporti visivi), un settore non ancora diventato *mainstream* ma a mio avviso molto promettente in ottica sanitaria, militare e industriale dato il ridotto movimento fisico e le tempistiche richieste per ottenere il contenuto informativo;
- Sperimentazione: al termine delle attività di tirocinio, ho potuto provare ad integrare il prodotto sviluppato con la libreria di componenti grafici *DevExtreme*; ho potuto riferire al responsabile del tirocinio un parere personale in merito alla semplicità di adozione e all'apporto che essa potrebbe dare ai prodotti aziendali in base al suo uso.



Full Name	Position	Hire Date	Address	Productivity Index
<input type="checkbox"/> John Heart	CEO	1/15/1995	351 S Hill St.	
<input type="checkbox"/> Sandra Johnson	COO	5/14/2012	3801 Chester Ave.	
<input type="checkbox"/> Robert Reagan	Shipping Manager	11/8/2002	4 Westmoreland Pl.	
<input checked="" type="checkbox"/> Greta Sims	Shipping Assistant	4/23/1998	547 Pacific St.	
<input checked="" type="checkbox"/> Brett Wade	Shipping Assistant	3/6/2009	1120 Old Mill Rd.	
<input type="checkbox"/> Olivia Peyton	Shipping Assistant	5/11/2005	4600 N Virginia Rd.	
<input type="checkbox"/> Cynthia Stanwick	CTO	8/11/2009	424 N Main St.	
<input type="checkbox"/> Kevin Carter	IT Manager	3/24/2008	2211 Bonita Dr.	
<input type="checkbox"/> Kent Samuelson	Support Manager	4/22/2009	12100 Mora Dr	
<input type="checkbox"/> Kevin Carter	Support Manager	3/24/2008	3801 Chester Ave.	
<input type="checkbox"/> Kent Samuelson	Support Manager	4/22/2009	4 Westmoreland Pl.	

Figura 1.6: Componente grafico *TreeList* di *DevExtreme*<sup>22</sup>

L'innovazione aziendale, dato l'ultimo punto del precedente elenco, è strettamente legata ai tirocini curriculari: il tempo a disposizione degli *stagisti* è usato anche per eseguire operazioni di prototipazione, valutando più o meno in profondità strumenti da poter adottare per futuri progetti e acquisendo un minimo di esperienza nell'ambito.

<sup>20</sup>Innovazione

<sup>21</sup>Realtà aumentata

<sup>22</sup>Fonte: <https://js.devexpress.com>



## Capitolo 2

# Motivazioni alla base del tirocinio

Questo capitolo si occupa di definire le motivazioni che hanno portato al compimento del percorso di tirocinio curricolare, dal punto di vista aziendale (si propone quella che, a mio parere, è la visione di *Trizeta*) e dal mio punto di vista, descrivendo i vincoli, gli obiettivi e le esigenze che il progetto proposto punta a soddisfare.

### 2.1 Strategia aziendale

In base a quanto ho potuto osservare e capire durante il periodo di *stage*, la strategia di gestione dei tirocini curricolari dell'azienda ospitante persegue i seguenti obiettivi:

- **Innovazione:** come riportato al termine della sezione §1.4, l'*innovazione* (ovvero l'introduzione di miglioramenti e novità) negli strumenti e nelle tecnologie utilizzate, motivata da esigenze produttive o di mercato, può avvalersi (come nel mio caso) del parere motivato del tirocinante tenendo conto del grado di maturità e delle caratteristiche del lavoro svolto per *testare* le novità da introdurre;
- **Integrazione di prodotti esistenti:** il parco *software Trizeta* è vasto (in relazione alle dimensioni dell'azienda) ed eterogeneo e la clientela richiede spesso l'implementazione di nuove funzionalità per rispondere a nuove esigenze; un modo per eseguire una prima integrazione di tali funzionalità (in versione *beta* o di *proof of concept*) si basa sul lavoro di uno o più tirocinanti;

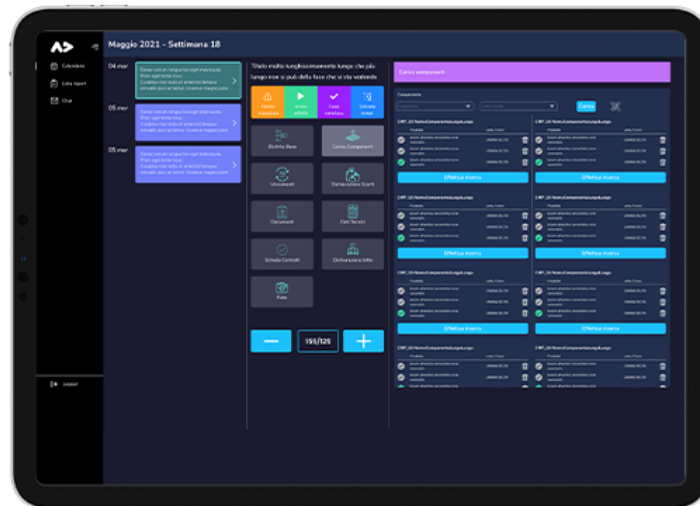


Figura 2.1: Interfaccia del software ADeMES *Trizeta*<sup>23</sup>

- **Creazione di nuovi prodotti:** è possibile che la risposta alle nuove esigenze della clientela non sia possibile direttamente all'interno dei prodotti già esistenti (per separazione di ambito o per non compromettere la mantenibilità dei prodotti esistenti): in questo caso, concordando lo *stack tecnologico* (l'insieme delle tecnologie) da usare, ho potuto implementare un prodotto prototipale *ex-novo* in base alle indicazioni ricevute;
- **Valutazione delle competenze** del tirocinante: il periodo di tirocinio è occasione di introduzione del tirocinante nel contesto aziendale e, per quanto riguarda la mia esperienza, formazione sulla visione aziendale, sui processi in atto, sulle tecnologie utilizzate e sulle abilità richieste per l'esecuzione delle attività lavorative; l'azienda ospitante fornisce quindi una formazione di base e valuta quali sono le reazioni agli stimoli (lavorativi e non) in funzione dell'inserimento del tirocinante nel *team* aziendale.

<sup>23</sup>Fonte: <https://trizeta.com>

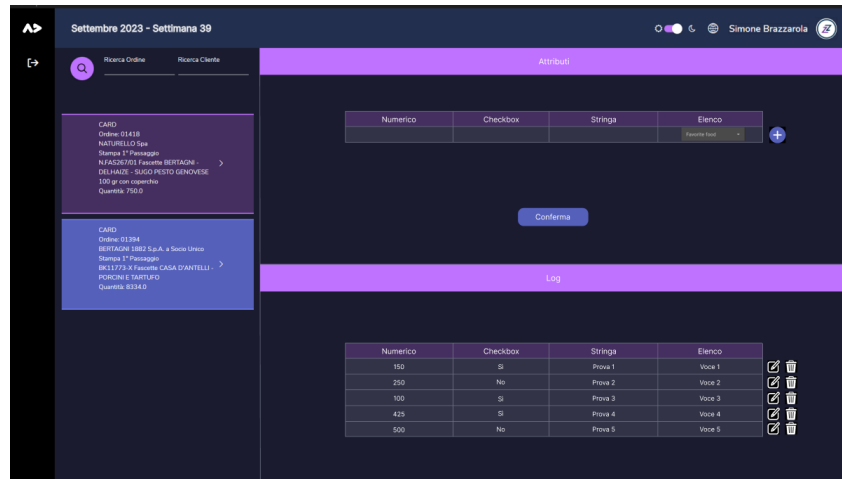
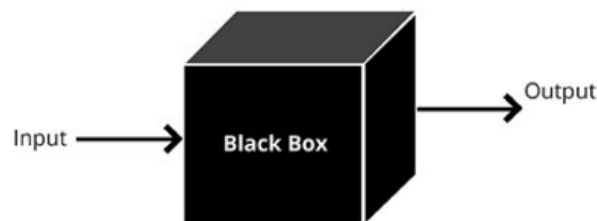


Figura 2.2: Interfaccia del prodotto ADeQA, oggetto del tirocinio

## 2.2 Problematiche poste in essere

Scopo delle mie attività di *stage* è la creazione di una *Progressive Web App*, ovvero un'applicazione *web* che mette a disposizione funzionalità aggiuntive rispetto a quelle offerte da un sito *web* (ad esempio, la fruizione dei servizi anche in modalità *offline*) per la raccolta di dati relativi al controllo della qualità delle linee produttive di aziende manifatturiere.

Ogni linea produttiva è costituita da un insieme di "fasi" di lavorazione: ogni fase di lavorazione è considerabile come una *black-box*<sup>24</sup>, ovvero un processo di cui si conoscono gli *input* (materia prima o semilavorati in ingresso) e gli *output* (semilavorati o prodotti finiti) ma non il funzionamento interno.

Figura 2.3: Schema sintesi del concetto di *black-box*<sup>25</sup>

Ogni fase è associata a un insieme di caratteristiche (dette "attributi") relative ai prodotti in uscita dalla stessa: esse sono di interesse per comprendere se la lavorazione ha prodotto articoli utilizzabili nelle fasi successive / nello stoccaggio degli stessi o meno.

Non è noto a priori il numero nè il tipo di attributi associati a una determinata fase di lavorazione: sono tutte informazioni ottenibili tramite l'utilizzo di servizi *backend*<sup>26</sup>

<sup>24</sup> *Black box*

<sup>25</sup> Fonte: <https://www.quora.com>

<sup>26</sup> *Backend*

(relativi alla struttura ed alla logica di persistenza) esposti dal *tutor* aziendale con una serie di *API*<sup>27</sup> *REST*<sup>28</sup> (astrazioni che consentono di eseguire operazioni sui dati di dominio in una architettura *client - server*, con la possibilità di salvare le risposte ottenute, senza memorizzare lo stato del software *client*).

L'obiettivo del prodotto è consentire all'utente (operatore in linea di produzione) di inserire, modificare, eliminare e visualizzare dati di controllo qualità per le fasi desiderate (in gergo tecnico, eseguire le operazioni *CRUD*, ovvero "*Create*", "*Read*", "*Update*" e "*Delete*", sui dati di qualità).

## 2.3 Obiettivi

Riporto le notazioni utilizzate in seguito per identificare gli obiettivi delle attività:

- **O** per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- **D** per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- **F** per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

Di seguito, la lista degli obiettivi di tirocinio:

- **Obbligatori**
  - **O01**: comprensione dei requisiti utente da soddisfare;
  - **O02**: studio dell'interfaccia dell'applicazione *ADeMES*, che verrà integrata con il prodotto da sviluppare durante il tirocinio;
  - **O03**: acquisizione della sufficiente dimestichezza con i concetti di base del *framework Angular*;
  - **O04**: progettazione dell'interfaccia grafica in base allo stile dell'interfaccia dell'applicazione *ADeMES*;
  - **O05**: sviluppo di una versione di base dell'applicazione *web* che consenta di eseguire le operazioni *CRUD* sui dati di qualità;
  - **O06**: *live demo* della web application in un ambiente simulato;
  - **O07**: studio e scelta (motivata) della tecnologia per la fruizione dell'applicazione in lingua inglese;
  - **O08**: il *software* deve potersi integrare nel software *ADeMES* mediante un elemento `<iframe>` *HTML*;
  - **O09**: il *software* deve poter essere eseguibile in modalità *standalone*<sup>29</sup> (in questo caso, in grado di funzionare anche senza l'ausilio del *software ADeMES*).

---

<sup>27</sup> [Application Program Interface](#)

<sup>28</sup> [Representational State Transfer](#)

<sup>29</sup> [Standalone](#)

- **Desiderabili**
  - **D01**: ottimizzazione dei servizi esposti.
- **Facoltativi**
  - **F01**: ottimizzazione dell'esperienza utente per compatibilità con *ADeMES*;
  - **F02**: ottimizzazione dell'interfaccia grafica, per rendere quanto più simile il prodotto a *ADeMES*;
  - **F03**: possibilità di fruizione dell'applicazione in lingua spagnola.

## 2.4 Vincoli

Il progetto si focalizza sullo sviluppo di codice relativo all'interfaccia grafica e questo aspetto caratterizza le condizioni imposte per lo svolgimento del lavoro e le aspettative sul risultato:

- Il prodotto deve essere sviluppato usando il *framework Angular* alla versione 16;
- Il prodotto deve essere una *Progressive Web App* (applicazione *web* installabile come se fosse un'applicazione nativa);
- Il prodotto deve fare uso delle *API* messe a disposizione da *Trizeta*, ovvero delle regole di comunicazione tra *frontend* (interfaccia grafica) e *backend* (*software di modellazione e memorizzazione dei dati di dominio*);
- Il prodotto deve essere utilizzabile dagli operatori di linea produttiva tramite dispositivi mobili (obbligatoriamente tramite *tablet*, facoltativamente tramite *smartphone*);
- Deve essere redatto un manuale utente che descriva interfaccia grafica ed esperienza utente del *software* sviluppato.

## 2.5 Pianificazione

Ho rispettato la pianificazione delle attività, redatta anticipatamente rispetto all'inizio del progetto e disponibile di seguito, per la maggior parte richiesto dalle attività di tirocinio: solamente l'interfacciamento coi servizi di scrittura dati è avvenuto in ritardo, data l'assenza temporanea degli stessi.

- **Prima Settimana (40 ore)**
  - Incontro con persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare;
  - Verifica credenziali e strumenti di lavoro assegnati;
  - Presa visione dell'infrastruttura esistente;

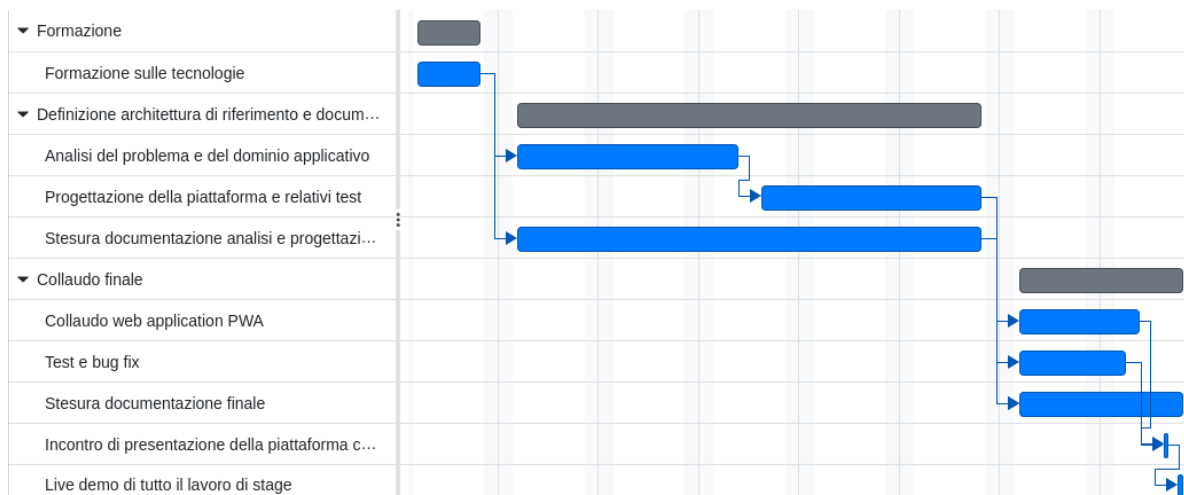
- Formazione sulle tecnologie adottate.
- **Seconda Settimana - (40 ore)**
  - Analisi e mappatura dei servizi esistenti;
  - Documentazione dell'analisi dei servizi esposti.
- **Terza Settimana - (40 ore)**
  - Analisi interfaccia/esperienza utente della *web application*;
  - Documentazione dell'analisi dell'interfaccia della *web application*;
  - Preparazione di un prototipo dimostrativo.
- **Quarta Settimana - (40 ore)**
  - Scelta della tecnologia e del *framework* da utilizzare;
  - Conclusione della documentazione di analisi;
  - Documentazione delle scelte progettuali;
  - Predisposizione infrastruttura della *web application*.
- **Quinta Settimana - (40 ore)**
  - Documentazione delle scelte progettuali;
  - Sviluppo *web application* ed interfacciamento con i servizi di lettura dati esposti.
- **Sesta Settimana - (40 ore)**
  - Conclusione della documentazione delle scelte progettuali;
  - Sviluppo *web application* ed interfacciamento con i servizi di scrittura dati esposti.
- **Settima Settimana - (40 ore)**
  - Collaudo dell'applicazione;
  - *Test* e correzione degli eventuali errori;
  - Stesura della documentazione finale.
- **Ottava Settimana - Conclusione (40 ore)**
  - Conclusione del collaudo dell'applicazione;
  - Conclusione *test* e correzione degli eventuali errori;
  - Conclusione della stesura della documentazione finale.

Di seguito, la distribuzione delle ore lavorative in relazione alle macro-attività:

Durata in ore	Descrizione dell'attività
40	Formazione sulle tecnologie
200	Definizione architettura di riferimento e relativa documentazione
-----	
	Analisi del problema e del dominio applicativo
	Progettazione della piattaforma e relativi <i>test</i>
	Stesura documentazione relativa ad analisi e progettazione
80	Collaudo Finale
-----	
	Collaudo
	Stesura documentazione finale
	Incontro di presentazione della piattaforma con gli <i>stakeholders</i>
	<i>Live demo</i> di tutto il lavoro di <i>stage</i>
<b>Totale ore</b>	<b>320</b>

Tabella 2.1: Distribuzione delle ore di lavoro

Riporto di seguito il diagramma di *Gantt* relativo al piano di lavoro previsto.

Figura 2.4: Diagramma *Gantt* di suddivisione delle attività

## 2.6 Scelta del tirocinio

Le ragioni che mi hanno portato a scegliere il progetto di tirocinio curricolare offerto da *Trizeta* possono essere così schematizzate:

- **Introduzione al mondo del lavoro (ambito informatico):** nonostante i tirocini curricolari sostenuti nel corso della mia carriera scolastica, prima di questo *stage* non avevo mai avuto occasione di lavorare in ambito informatico, probabilmente per timore di non essere in grado di affrontare le sfide dello sviluppo di prodotti *software*;
- **Desiderio di conoscenza di una realtà locale:** avendo frequentato un istituto di istruzione superiore limitrofo alla sede aziendale, ho sempre sentito

nominare *Trizeta* dai docenti e dai compagni di studi dato il suo coinvolgimento in attività didattiche organizzate dall'istituto scolastico;

- **Curiosità per lo sviluppo *frontend*:** nel corso della mia esperienza universitaria, la progettazione e la realizzazione di interfacce grafiche sono state piuttosto scarse e non si sono mai confrontate con vere e proprie richieste da parte di utenti finali o persone interessate nella buona riuscita del progetto (*stakeholders*) che avessero ben chiari i bisogni e le esigenze a cui dare risposta.

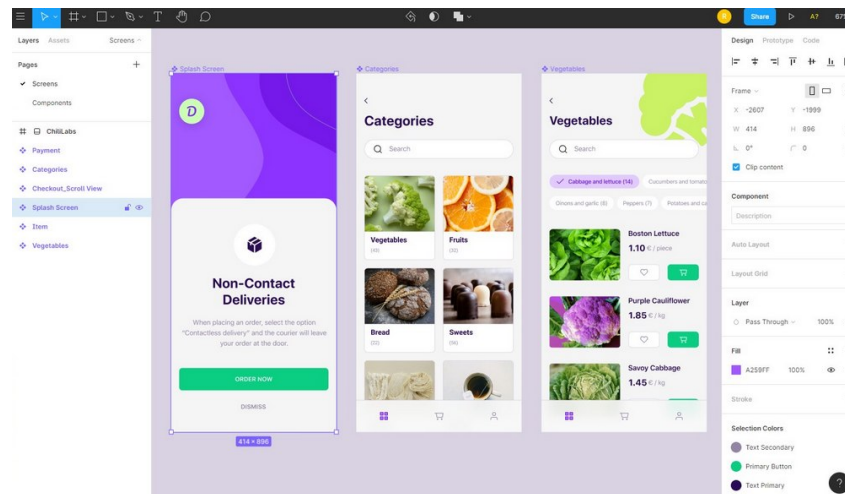


Figura 2.5: Interfaccia del *software Figma*, usato per la progettazione dell'interfaccia utente<sup>30</sup>

Queste motivazioni (ed il contesto personale in cui ho svolto le attività lavorative) mi hanno consentito di definire una serie di obiettivi personali per quanto riguarda la scelta dell'azienda, il tema delle attività e l'acquisizione di conoscenze e competenze:

- Capire come convertire una *web application* in una *Progressive Web App*, ovvero un'applicazione che coniuga caratteristiche di un'applicazione nativa (quali l'installazione e l'aspetto grafico) e caratteristiche di un'applicazione *web* (come la possibilità di utilizzo tramite *browser*);
- Sviluppare un'interfaccia grafica che si adatti a *desktop*, *tablet* e *smartphone*;
- Comprendere come rendere fruibile in più lingue un prodotto *software*;
- Capire come si possono gestire diversi temi grafici (tipicamente identificati come "tema chiaro" e "tema scuro") in un'interfaccia grafica *web*;
- Ideare un prodotto in grado di integrarsi con successo in un *software* già esistente.

<sup>30</sup>Fonte: <https://www.toponseek.com>



## Capitolo 3

# Elementi caratterizzanti del progetto

Questo capitolo si occupa di introdurre gli strumenti utilizzati nel corso del tirocinio, definire e dare una visione concreta delle attività di sviluppo del progetto di *stage* e dare prova dei risultati raggiunti a livello di documentazione, di codice scritto, *test coverage* e obiettivi raggiunti.

### 3.1 Stile lavorativo

Data la pianificazione a cadenza settimanale delle attività (sezione §2.5), ho concordato con il *tutor* aziendale, il signor *Michele Rigo*, l'organizzazione di una riunione di allineamento settimanale, programmata per ogni lunedì mattina.

Tale incontro mirava a:

- Mostrare il lavoro svolto nel corso della settimana precedente alla riunione;
- Condurre una retrospettiva sulla settimana precedente;
- Valutare lo stato di avanzamento del progetto in relazione alle aspettative;
- Delineare le attività da svolgere nella settimana in corso.

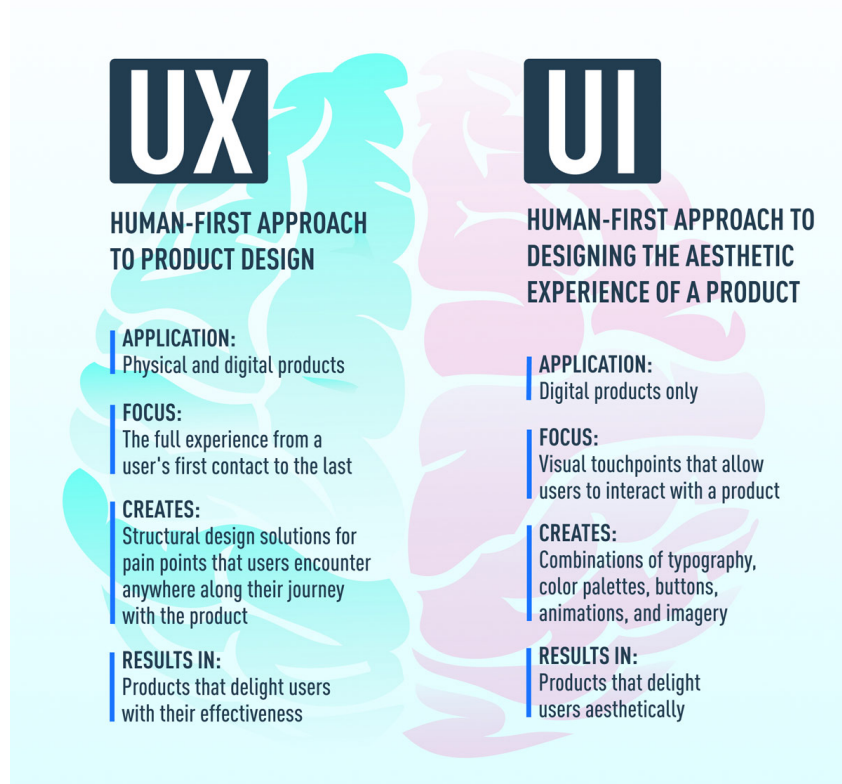


Figura 3.1: Confronto tra la progettazione di interfacce (*UI, User Interface*) e di esperienze utente (*UX, User Experience*)<sup>31</sup>

Abbiamo svolto degli incontri aggiuntivi in forma telematica per chiarire alcuni aspetti del progetto, in particolare la comprensione di alcuni requisiti utente e la definizione di determinati aspetti di interfaccia grafica ed esperienza utente (ovvero come l'utente può interagire con l'interfaccia grafica, la relazione che intercorre tra egli e gli elementi grafici).

Per tutta la durata delle attività di *stage*, sono stato affiancato in caso di necessità da praticamente tutto il team *Trizeta*, soprattutto per quanto concerne la progettazione dell'interfaccia grafica.

## 3.2 Strumenti utilizzati

### 3.2.1 Strumenti di sviluppo

#### Angular

Versione: 16.2.9

*Descrizione:* è un *framework open-source* per lo sviluppo di applicazioni *web* a singola pagina (è un tipo di applicazione *web* che opera all'interno di una singola pagina *web*, senza la necessità di ricaricarla durante l'interazione dell'utente).

Sviluppato da *Google*, *Angular* fornisce una struttura per la costruzione di

<sup>31</sup>Fonte: <https://careerfoundry.com>

applicazioni *web* che consentono agli sviluppatori di utilizzare il linguaggio *TypeScript* o *JavaScript* per la creazione di componenti riutilizzabili.

### Angular Material

*Versione:* 16.2.8

*Descrizione:* è una libreria di componenti grafiche e direttive, sviluppata da *Google* e progettata per essere utilizzata con il *framework Angular*.

Questa libreria fornisce una serie di componenti predefiniti e stilizzati che semplificano la creazione di interfacce utente coerenti e moderne all'interno delle applicazioni *Angular*.

### CSS

*Versione:* 3

*Descrizione:* è un linguaggio di stile utilizzato per definire la presentazione di documenti *HTML* e *XML*; determina come i documenti devono essere visualizzati sullo schermo, sulla carta o in altri tipi di supporto.

### Figma

*Versione:* 9.0

*Descrizione:* è un'applicazione di progettazione e prototipazione basata su *cloud* (*software* il cui funzionamento e archiviazione dei dati avvengono prevalentemente attraverso risorse di calcolo e archiviazione disponibili su *Internet*, anziché su risorse locali o *server* fisici) che consente di collaborare in tempo reale su progetti di interfaccia utente (*UI*) ed esperienza utente (*UX*).

### HTML

*Versione:* 5

*Descrizione:* è il linguaggio *standard* utilizzato per la creazione e la strutturazione di pagine *web*; lavora in collaborazione con *CSS* e *JavaScript* per creare esperienze *web* complete e interattive.

### Jasmine

*Versione:* 4.6.0

*Descrizione:* è un *framework* di *testing* popolare per *JavaScript* ed è comunemente utilizzato per testare le applicazioni *Angular*.

### Karma

*Versione:* 6.4.0

*Descrizione:* è uno strumento ampiamente utilizzato per l'esecuzione di *test* di unità per applicazioni *Angular*.

*Karma* genera un *server web* che esegue il codice di *test Javascript* (e *TypeScript*, che ne è sovralinguaggio) per ogni *browser* connesso.

### Node.js

*Versione:* 18.17.1

*Descrizione:* è un ambiente di *runtime open source* basato sul motore *JavaScript V8* di *Google Chrome*.

Consente di eseguire codice *JavaScript* lato *server*, dando l'opportunità agli sviluppatori di utilizzare *JavaScript* per lo sviluppo di applicazioni *back-end*.

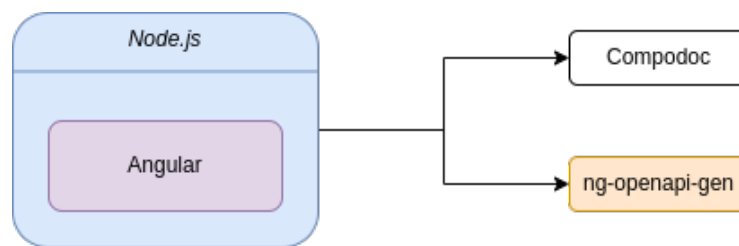


Figura 3.2: Diagramma del *runtime Node.js* e dei pacchetti in esso installati

### Ng-openapi-gen

*Versione:* 0.50.2

*Descrizione:* è un modulo *npm* (il gestore di pacchetti per *Node.js*) che genera servizi, modelli e funzioni *Angular* a partire da una specifica *OpenAPI 3* (è uno *standard* che aiuta a descrivere e documentare le Interfacce di Programmazione delle Applicazioni, [API](#)).

### Ngx-translate

*Versione:* 15.0.0

*Descrizione:* libreria che consente l'internazionalizzazione (ovvero facilita l'astrazione del contenuto statico di un'applicazione *web* rispetto alla lingua di fruizione) e la localizzazione (ovvero consente di adattare il *software* in base alle esigenze culturali dell'area in cui il prodotto viene usato) in *Angular*.

### StarUML

*Versione:* 6.0.1

*Descrizione:* è uno strumento di modellazione *UML*<sup>32</sup> (*Unified Modeling Language*) che offre un ambiente grafico per progettare e visualizzare diagrammi *UML*. *UML* è uno *standard* per la modellazione visuale di sistemi *software*, ed è

---

<sup>32</sup> *Unified Modeling Language*

utilizzato per rappresentare graficamente diversi aspetti di un sistema come le classi, i casi d'uso, le sequenze di chiamate, le attività, e altro ancora.

### TypeScript

Versione: 5.1.6

*Descrizione:* è un linguaggio di programmazione *open-source* sviluppato da *Microsoft*. È una versione "*superset*" di *JavaScript*, il che significa che aggiunge nuove funzionalità e tipizzazione statica al linguaggio *JavaScript*.

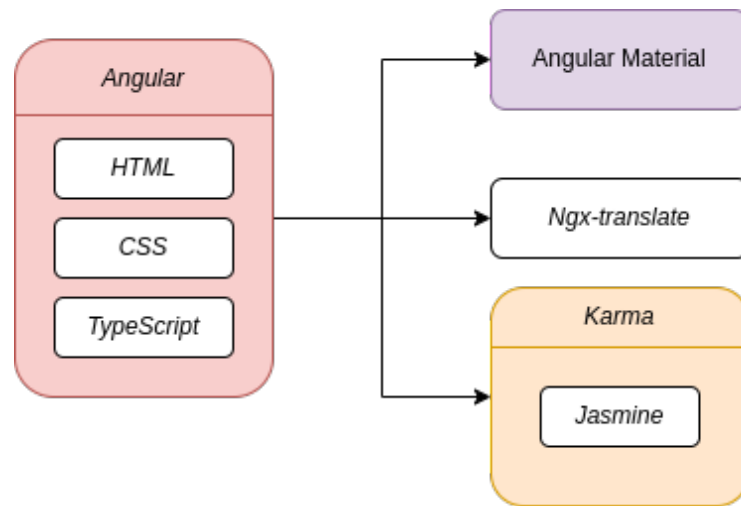


Figura 3.3: Diagramma dei *framework*, delle librerie e dei linguaggi usati

### Visual Studio Code

Versione: 1.84.1

*Descrizione:* è un *editor* di codice sorgente gratuito e *open-source* sviluppato da *Microsoft*.

È progettato per essere leggero, flessibile e altamente personalizzabile, rendendolo uno strumento popolare tra gli sviluppatori per la scrittura di codice in diversi linguaggi di programmazione.

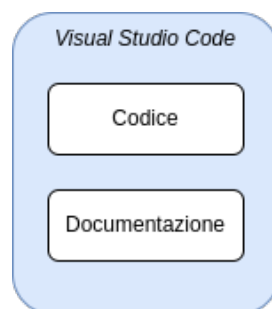


Figura 3.4: Diagramma degli *editor* di testo usati

### 3.2.2 Strumenti di versionamento

#### Git

*Versione:* 2.43.0

*Descrizione:* è un sistema di controllo delle versioni distribuito (*DVCS*, *Distributed Version Control System*), utilizzato per tracciare le modifiche apportate al codice sorgente durante lo sviluppo del *software*.

#### GitHub

*Versione:* 3.11.0

*Descrizione:* è una piattaforma di *hosting* per il controllo delle versioni e la collaborazione.

Offre servizi basati su *Git* e facilita la gestione e la condivisione dei progetti *software*.

### 3.2.3 Strumenti di documentazione

#### Compodoc

*Versione:* 1.1.22

*Descrizione:* strumento *open-source* per la generazione di documentazione per *web app Angular* a partire da commenti scritti nel codice sorgente.

#### LibreOffice

*Versione:* 7.6.2

*Descrizione:* è una *suite* di *software* per l'ufficio libera e *open-source* che offre un insieme di applicazioni per la produttività personale e professionale.

È sviluppato dalla comunità di sviluppatori di *The Document Foundation* ed è una delle alternative più popolari e complete a *suite* di produttività come *Microsoft Office*.

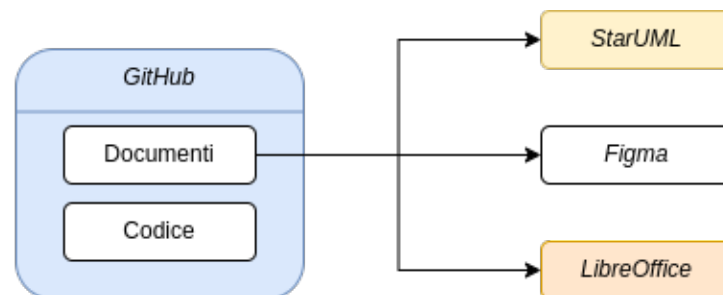


Figura 3.5: Diagramma che rappresenta la relazione tra la documentazione (gestita tramite *GitHub*) e gli strumenti adottati per redigerla

### 3.3 Analisi dei requisiti

Le attività di **analisi dei requisiti** hanno come scopo la comprensione dei bisogni espressi dall'utente finale (o da chi vuole risolvere tali bisogni all'utente finale, talvolta nemmeno noti all'utente finale <sup>33</sup>) detti "requisiti utente", e la definizione di "ciò che deve svolgere il prodotto" per soddisfare tali bisogni, detti "requisiti del prodotto". Ho iniziato le attività di **analisi** con un incontro sincrono, insieme al *tutor* aziendale, relativo all'infrastruttura aziendale esistente (contesto delle attività produttive) e al ruolo che il prodotto da sviluppare (di nome *ADeQA*) ricopre in questo contesto. In seguito alla contestualizzazione delle mie attività di *stage*, il *tutor* aziendale ed io abbiamo discusso dei bisogni da soddisfare: ho dovuto eseguire più passi di raffinazione dei bisogni espressi (descritti in seguito) per arrivare a definire i requisiti del prodotto.

#### 3.3.1 Casi d'uso

Con "caso d'uso" si intende un'astrazione utilizzata per catturare, descrivere e definire le interazioni tra un prodotto e gli attori (utenti o altri *software*) che interagiscono con esso.

Un caso d'uso è una rappresentazione (narrativa e/o schematica) di uno scenario che descrive come il sistema deve rispondere alle richieste degli utenti in determinate circostanze: serve per eseguire una prima schematizzazione dei bisogni utente, specificandoli in una serie casi d'uso più dettagliati qualora ve ne fosse la possibilità.

I casi d'uso qui presenti mirano a dare comprensione dei requisiti utente principali, pertanto non riporto tutti i casi d'uso identificati in sede di *stage*.

#### Nomenclatura

I casi d'uso sono identificati da una sigla alfanumerica così composta:

**UC[Tipologia]-[Codice]**

- **UC**: abbreviativo di "*Use Case*";
- **Tipologia**: tipologia del caso d'uso:
  - **F**: funzionale, descrive una funzionalità;
  - **E**: errore, descrive cosa deve accadere in caso di un determinato errore.
- **Codice**: identificativo numerico del caso d'uso, può identificare dei sotto-casi d'uso / generalizzazioni qualora si presentasse in forma **[caso].[identificativo]**.

#### Attori primari

L'applicazione presenta due attori primari, ovvero due tipologie di utente finale:

- **Utente non autenticato**: utente che non ha ancora effettuato l'autenticazione, avrà funzionalità limitate rispetto ad un utente autenticato;

---

<sup>33</sup>Fonte: <https://www.mountaingoatsoftware.com>

- **Utente autenticato:** utente che ha effettuato l'autenticazione alla piattaforma tramite le proprie credenziali (nome utente, *password* e *pin*); ha accesso ad ogni funzionalità messa a disposizione.

### Lista dei principali casi d'uso

#### UCF-2: Visualizzazione delle fasi

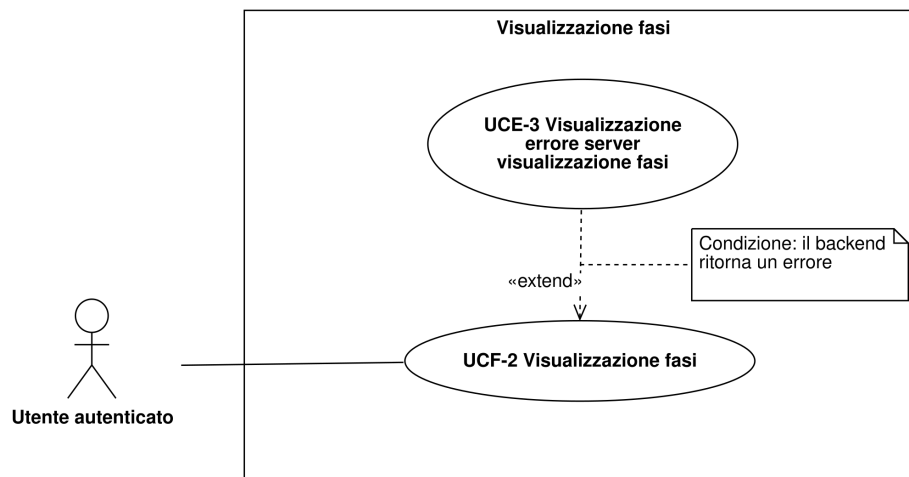


Figura 3.6: Diagramma dei casi d'uso UCF-2 e UCE-3.

- **Descrizione:** l'utente vuole visualizzare le fasi di lavorazione della filiera produttiva;
- **Scenario:**
  1. L'utente visualizza le fasi di lavorazione.
- **Estensioni:** si presenta un errore lato server alla lettura delle fasi di produzione (**UCE-3**);
- **Attore principale:** utente autenticato;
- **Precondizioni:** l'utente è autenticato;
- **Postcondizioni:** l'utente visualizza le fasi di produzione.

Questo caso d'uso schematizza la relazione che intercorre tra l'applicazione da sviluppare e l'utente (in questo caso, "utente autenticato") per quanto riguarda la visualizzazione delle fasi di lavorazione: questa relazione non è ulteriormente scomponibile in ulteriori sotto-relazioni (tramite sotto-casi d'uso) nè tantomeno è una generalizzazione, ovvero non è possibile individuare ereditarietà con altri casi d'uso.

#### UCF-5: Valorizzazione degli attributi



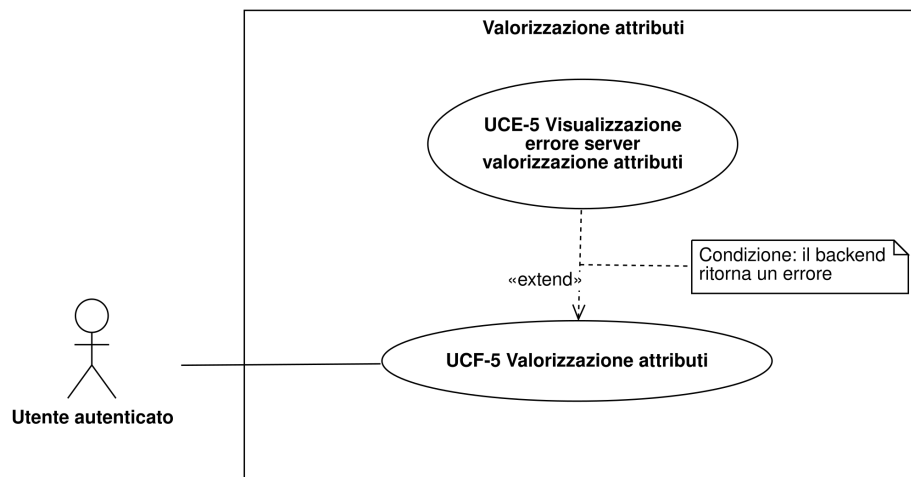


Figura 3.7: Diagramma dei casi d'uso UCF-5 e UCE-5.

- **Descrizione:** l'utente vuole inserire manualmente un valore per gli attributi della fase di lavorazione selezionata, creando un *log* di controllo qualità;
- **Scenario:**
  1. L'utente visualizza gli attributi della fase di lavorazione selezionata;
  2. L'utente assegna un valore agli attributi visualizzati;
  3. L'utente conferma la scelta.
- **Estensioni:** si presenta un errore lato server all'inserimento del valore degli attributi della fase di produzione selezionata (**UCE-5**);
- **Attore principale:** utente autenticato;
- **Precondizioni:** l'utente sta visualizzando gli attributi (**UCF-4**);
- **Postcondizioni:** l'utente assegna un valore agli attributi di una fase di produzione che ha selezionato.

Questo caso d'uso schematizza la relazione che intercorre tra l'applicazione da sviluppare e l'utente autenticato relativamente all'inserimento di dati di controllo qualità: come per il caso precedente, non è scomponibile in ulteriori sotto-relazioni e non è possibile individuare una generalizzazione, dato che i dati di qualità da inserire non sono noti a priori in numero nè in tipo.

### 3.3.2 Requisiti

Con "requisito" si intende una specifica di ciò che un sistema software deve fare o possedere al fine di soddisfare un determinato obiettivo o risolvere un particolare problema; i requisiti sono le basi per la **progettazione** e la **verifica** di un *software*. La lista di requisiti (che riporto di seguito) offre una base per la comprensione degli esempi di **progettazione** e **codifica**.

I casi d'uso *UCF-2* e *UCF-5* sono associati rispettivamente ai requisiti *RF-2* e *RF-5* della tabella (parzialmente completa) dei [requisiti funzionali](#).

### Nomenclatura

I requisiti qui presenti sono identificati da una sigla alfanumerica così composta:

**R[Tipologia]-[Codice]**

- **R:** abbreviativo di "requisito";
- **Tipologia:** tipologia del requisito:
  - **F:** funzionale, descrive una funzionalità (ciò che un prodotto consente di fare all'utente);
  - **Q:** di qualità, indica una caratteristica delle modalità di sviluppo;
  - **P:** prestazionale, indica alcune caratteristiche che il prodotto deve avere durante l'esecuzione (non ciò che consente di fare);
  - **V:** di vincolo, ovvero riguardante una caratteristica del prodotto stabilita prima del suo sviluppo.
- **Codice:** identificativo numerico del requisito, può identificare dei sotto-requisiti qualora si presentasse in forma **[requisito].[sotto-requisito]**; i requisiti funzionali relativi a casi d'uso di errore presentano forma **E[requisito].[sotto-requisito]**.

### Requisiti funzionali

Codice	Descrizione	Classificazione
RF-1	L'utente deve poter inserire i propri dati per effettuare il <i>login</i>	Obbligatorio
RF-1.1	L'utente deve poter inserire la coppia "nome utente, <i>password</i> " per effettuare il <i>login</i>	Obbligatorio
RF-1.2	L'utente deve poter inserire il <i>pin</i> per effettuare il <i>login</i>	Obbligatorio
RF-2	L'utente deve poter visualizzare le fasi di produzione della propria filiera	Obbligatorio
RF-3	L'utente deve poter filtrare le fasi di produzione della propria filiera	Desiderabile
RF-3.1	L'utente deve poter filtrare le fasi di produzione della propria filiera in base al codice dell'ordine	Desiderabile
RF-3.2	L'utente deve poter filtrare le fasi di produzione della propria filiera in base al nome del cliente	Desiderabile
RF-4	L'utente deve poter visualizzare gli attributi della fase di produzione selezionata	Obbligatorio
RF-5	L'utente deve poter assegnare un valore agli attributi della fase di produzione selezionata	Obbligatorio
RF-5.1	L'utente deve poter visualizzare un messaggio positivo in caso l'assegnazione di valori agli attributi avvenga correttamente	Obbligatorio

Tabella 3.1: Requisiti funzionali

## Requisiti di qualità

Codice	Descrizione	Classificazione
RQ-1	Deve essere fornito un manuale utente per l'utilizzo	Obbligatorio
RQ-2	Il codice sorgente deve essere presente in <i>GitHub</i>	Obbligatorio

Tabella 3.2: Requisiti di qualità

## Requisiti prestazionali

Codice	Descrizione	Classificazione
RP-1	L'interfaccia utente deve adattarsi a dispositivi <i>tablet</i>	Obbligatorio
RP-2	L'interfaccia utente deve adattarsi a smartphone	Facoltativo

Tabella 3.3: Requisiti prestazionali

## Requisiti di vincolo

Codice	Descrizione	Classificazione
RV-1	L'applicazione deve essere sviluppata usando il <i>framework Angular</i> alla versione 16	Obbligatorio
RV-2	L'applicazione deve sfruttare i servizi messi a disposizione dal <i>software</i> di <i>backend</i> aziendale	Obbligatorio
RV-3	L'applicazione deve essere integrabile in un <code>&lt;iframe&gt;</code> <i>HTML</i> dell'applicativo <i>ADeMES</i> ( <i>suite</i> aziendale)	Obbligatorio
RV-4	L'applicazione deve essere una <a href="#">Progressive Web App</a>	Obbligatorio
RV-5	L'applicazione deve essere fruibile in inglese	Obbligatorio
RV-6	L'applicazione deve essere fruibile in spagnolo	Facoltativo

Tabella 3.4: Requisiti di vincolo

## Riepilogo

Di seguito, due tabelle riassuntive relative a:

- Classificazione dei requisiti;
- Numero di requisiti individuati per ogni tipologia.

Classificazione	Quantità
Obbligatori	32
Desiderabili	3
Facoltativi	2

Tabella 3.5: Tabella riassuntiva delle classificazioni dei requisiti

Tipologia	Quantità
Funzionali	27
Di qualità	2
Prestazionali	2
Di vincolo	6

Tabella 3.6: Tabella riassuntiva dei requisiti

## 3.4 Progettazione

Le attività di **progettazione** hanno come scopo l'ideazione della struttura del prodotto (a livello di programmazione, a livello di interfaccia, a livello di interazioni con l'utente e con altri sistemi) tramite delle scelte effettuate in base a determinati obiettivi (qualità del codice) ed ai requisiti individuati.

L'*output* della **progettazione** è costituito da uno o più documenti di specifica, ovvero documenti contenenti dettagli su come il *software* dovrebbe essere implementato.

Di seguito, si approfondiscono le scelte effettuate per:

- La fruizione in multipli linguaggi dell'applicazione;
- Definire l'interfaccia grafica;
- Specificare l'implementazione per i casi d'uso riportati nella sezione §3.3.1.

### 3.4.1 Internazionalizzazione e localizzazione

Per poter capire le scelte progettuali occorre prima dare delle definizioni ai seguenti termini:

- **Internazionalizzazione:** è il processo di **progettazione** e sviluppo di un prodotto in modo tale da ridurre la difficoltà di adattarlo per renderlo fruibile da persone appartenenti a culture diverse;
- **Localizzazione:** è il processo di adattamento di un prodotto in base alle esigenze culturali di una particolare area / di un particolare mercato.

Per quanto riguarda l'internazionalizzazione e la localizzazione del contenuto statico dell'applicazione, si è scelto di usare la libreria *ngx-translate* al posto del pacchetto *@angular/localize*, nativamente supportato da *Angular*.

Di seguito, una breve analisi sulle potenzialità e sulle carenze di entrambe le tecnologie, identificate tramite ricerche personali<sup>34 35 36</sup> e la costruzione di applicazioni *ad hoc*.

#### *@angular/localize*

Vantaggi :

- Supportato nativamente dal *framework Angular*;

<sup>34</sup><https://medium.com>

<sup>35</sup><https://stackoverflow.com>

<sup>36</sup><https://github.com>

- Velocità di esecuzione delle applicazioni che la usano;
- Supporto concreto per progetti di grandi dimensioni, data la scalabilità offerta (ovvero la capacità di gestire un aumento del carico di lavoro o delle risorse senza subire un degrado delle prestazioni).

Svantaggi :

- Necessita di una compilazione manuale per aggiornare le traduzioni;
- Necessita di un passaggio manuale di dati per aggiornare ogni traduzione;
- Presenza di multiple versioni dell'applicazione (una per ogni traduzione);
- Ogni versione necessita del proprio processo di compilazione;
- Ogni cambio di lingua si traduce nel caricamento di una versione diversa della stessa applicazione (con conseguente caricamento di una nuova pagina).

#### *ngx-translate*

Vantaggi :

- Di facile utilizzo e apprendimento;
- Utilizzo di file *JSON*, facilmente gestibili e modificabili;
- Consente di gestire la mancanza di traduzione;
- Consente un cambio di linguaggio di traduzione senza necessità di ricaricare la pagina.

Svantaggi :

- È una libreria nata come soluzione temporanea ai problemi di internazionalizzazione del *framework Angular* e quindi non è frequentemente aggiornata;
- Appesantisce l'esecuzione dell'applicazione.

La scelta di *ngx-translate* è stata dettata dalle ridotte dimensioni del prodotto e dalle sue esigenze, in particolare:

1. **Integrazione in un <iframe>**: dato che la *web app* deve essere in grado di eseguire all'interno di un <iframe> e l'applicazione "contenitore" utilizza tale libreria, si vuole garantire una traduzione uniforme all'interno della schermata visualizzata, evitando di ricaricare l'applicazione all'interno della porzione di schermo delimitata dall'<iframe>;
2. **Semplicità di sviluppo**: l'applicazione non ha dimensioni tali da prendere in considerazione misure di traduzione scalabili; questo consente di prediligere un approccio orientato alla semplicità di utilizzo durante lo sviluppo e la manutenzione del prodotto;
3. **Semplicità di test**: il tempo dedicato al *testing* delle traduzioni deve essere proporzionale all'apporto di tale funzionalità ed alle dimensioni dell'applicazione.

### 3.4.2 Interfaccia grafica

La **progettazione** dell'interfaccia grafica si è basata su colori e componenti grafiche già presenti nel *software ADeMES*: la vera problematica da risolvere in queste attività è stata l'adattamento dell'interfaccia grafica a dispositivi con schermo di dimensioni ridotte (*tablet* e *smartphone*).

#### Autenticazione - passo 1

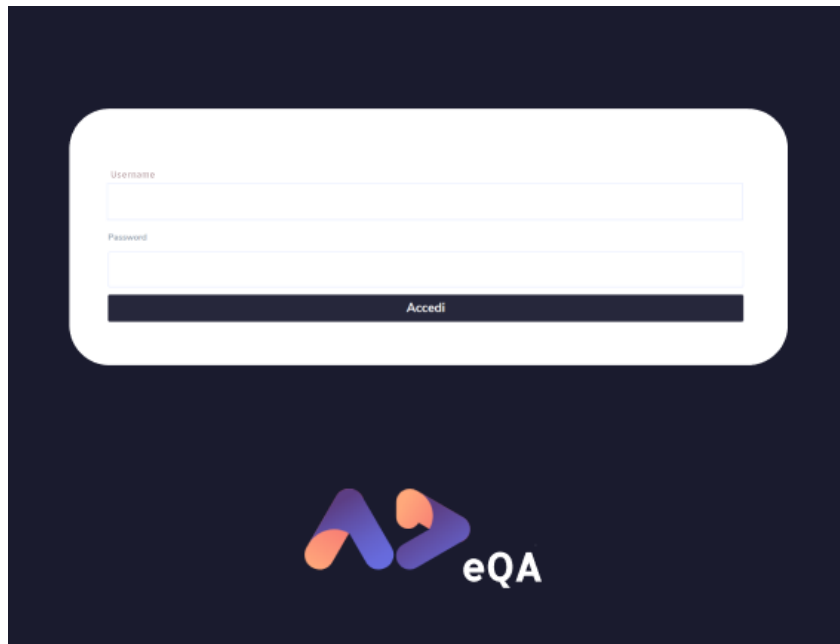


Figura 3.8: Interfaccia ideata per la prima schermata di autenticazione

L'interfaccia per il primo passo di accesso è minimale, essendo costituita dal logo dell'applicazione ed un *form* contenente due caselle di testo ed un bottone.

**Autenticazione - passo 2**

Figura 3.9: Interfaccia ideata per la seconda schermata di autenticazione

L'interfaccia per il secondo passo di accesso è molto simile all'interfaccia per il primo passo, con l'unica differenza di avere una "pulsantiera" per l'inserimento del *pin*; tale scelta è stata effettuata per compatibilità con l'interfaccia grafica del *software ADeMES*, nel quale *ADeQA* dovrà integrarsi.

## Visualizzazione principale

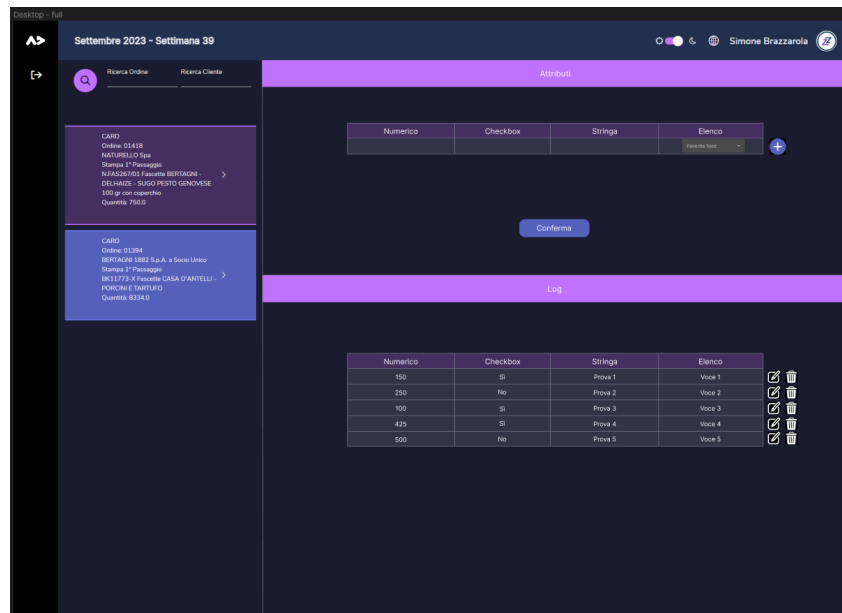


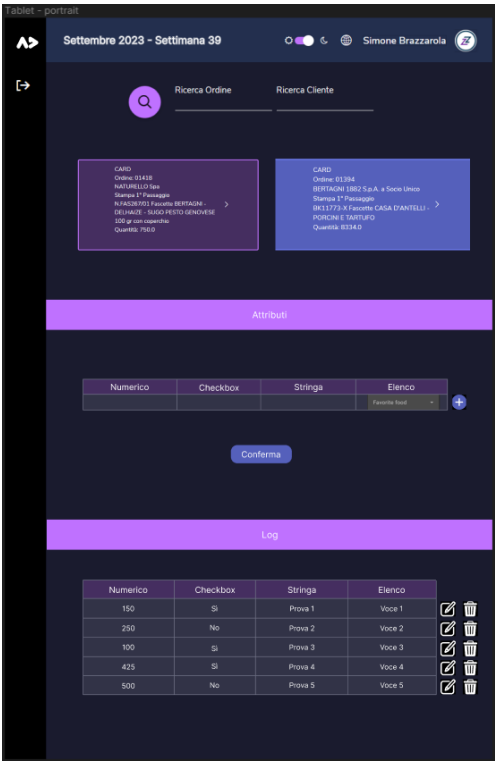
Figura 3.10: Interfaccia ideata per la visualizzazione principale

Questa pagina appare dopo l'autenticazione ed è contraddistinta dai seguenti elementi:

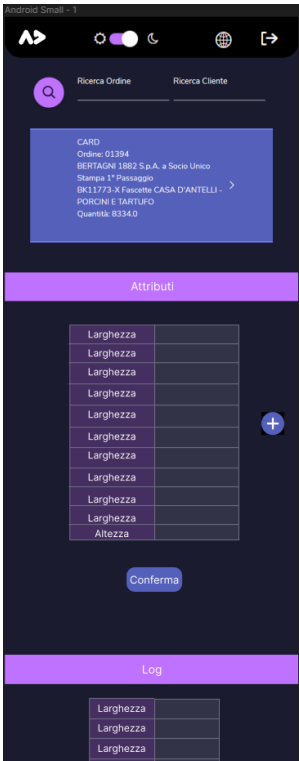
- **Menù centrale:** contiene l'indicazione della data attuale (mese, anno e settimana corrente), un *widget* per cambiare il tema dell'applicazione (chiaro / scuro), un *widget* per cambiare lingua all'applicazione, il nome utente dell'utente che ha eseguito l'accesso ed il logo aziendale;
- **Menù laterale:** contiene il logo della *suite* di prodotti aziendali ed un pulsante per eseguire il *logout*;
- **Barra laterale:** contiene una lista delle fasi di lavorazione attive per l'organizzazione di cui fa parte l'utente che ha fatto l'accesso all'applicazione (con scorrimento verticale in caso eccedessero lo spazio verticale a disposizione); in alto si trova uno strumento di ricerca / filtraggio atto a ridurre il numero di fasi visualizzate;
- **Cornice principale**
  - **Attributi:** questa porzione di schermo consente l'aggiunta e la modifica di informazioni relative al controllo qualità tramite una tabella avente numero di colonne variabile (in base al numero degli attributi);
  - **Log:** questa porzione di schermo consente la visualizzazione delle informazioni di controllo qualità già inserite per la fase selezionata (oltre alla possibilità di eliminare e indicare la volontà di modificare tali informazioni).



Visualizzazione per dispositivi mobili



(a) Visualizzazione principale - tablet



(b) Visualizzazione principale - smartphone

Figura 3.11: Visualizzazione principale adattata a dispositivi mobili

Visualizzazione con tema chiaro

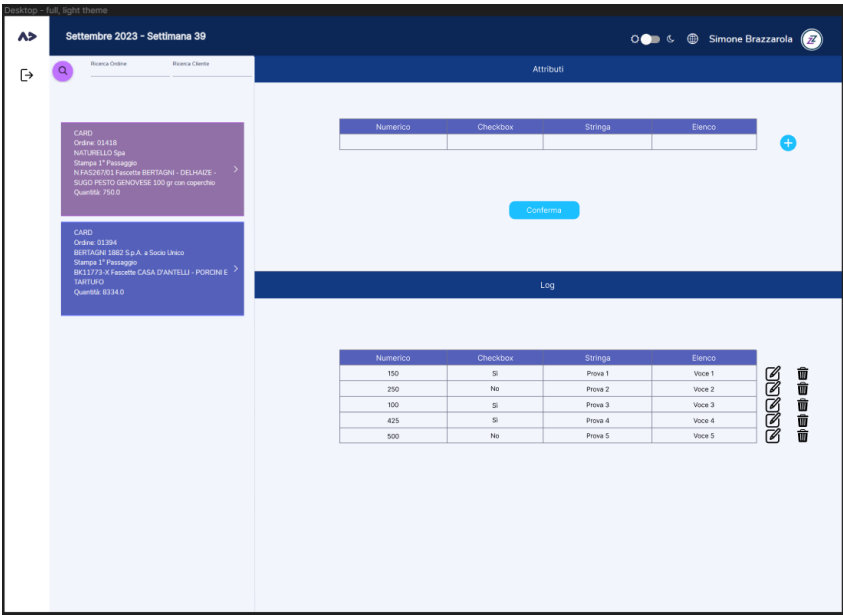


Figura 3.12: Visualizzazione principale con colori chiari

### 3.4.3 Architettura

L'architettura del prodotto si basa sui concetti di "servizio" e "componente" *Angular*:

- **Componente:** un componente del *framework Angular* è una parte riutilizzabile e autonoma dell'interfaccia utente, associata a una porzione specifica della pagina *web* o dell'applicazione.  
Un componente *Angular* è composto da un *file TypeScript* che definisce la logica del componente, un *file HTML* che definisce la struttura grafica, un *file* di stile *CSS* per la presentazione e un *file* di *test* per la verifica dell'implementazione;
- **Servizio:** un servizio del *framework Angular* è una classe *singleton* (ovvero della quale esiste una sola istanza in tutto il *software*, durante la sua esecuzione) che consente di avere funzionalità specifiche o dati condivisi all'interno di un'applicazione.  
I servizi in *Angular* sono ideati per fornire una separazione delle responsabilità e consentono di manipolare dati, facilitare la comunicazione tra componenti ed effettuare richieste a *software backend*.

Di seguito si riporta l'architettura delle classi relative ai casi d'uso fondamentali, indicati nella sezione §3.3.1.

#### Servizio di gestione delle informazioni riservate

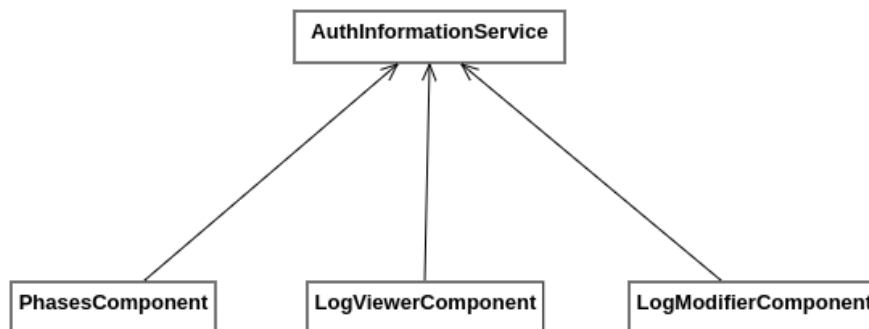


Figura 3.13: Diagramma delle classi del servizio di autenticazione

Il servizio *AuthInformationService* consente la gestione delle informazioni di autenticazione: nome utente, identificativo utente, tema grafico predefinito all'avvio e informazioni necessarie alla comunicazione con i servizi di *backend* esposti.

Classi relative a componenti *Angular*:

- *PhasesComponent*: si occupa della gestione delle fasi di lavorazione;
- *LogViewerComponent*: si occupa della visualizzazione dei dati di controllo qualità;
- *LogModifierComponent*: si occupa delle operazioni sui dati di controllo qualità salvati.

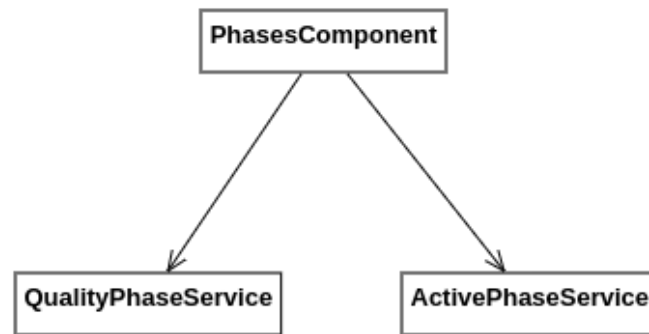
**Componente di gestione delle fasi**

Figura 3.14: Diagramma delle classi del componente di gestione delle fasi

Classi relative a servizi *Angular*:

- *QualityPhaseService*: servizio per l'ottenimento delle fasi di lavorazione;
- *ActivePhaseService*: servizio di gestione della fase selezionata dall'utente, serve per ottenere dinamicamente gli attributi per il controllo qualità.

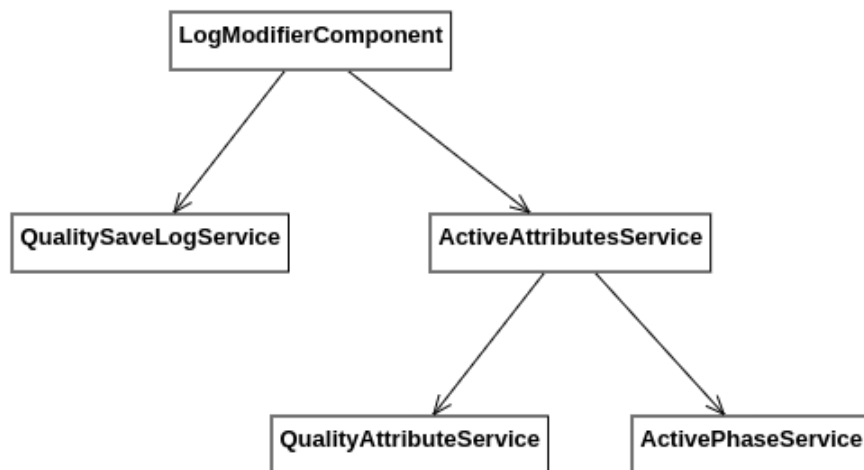
**Componente di modifica dei dati di controllo qualità**

Figura 3.15: Diagramma delle classi del componente di modifica dei dati di qualità

Classi relative a servizi *Angular*:

- *QualitySaveLogService*: servizio per l'esecuzione di operazioni sui dati di controllo qualità;

- *QualityAttributeService*: servizio per l'ottenimento degli attributi per la fase di lavorazione attiva;
- *ActiveAttributesService*: servizio di gestione degli attributi del controllo qualità per la fase selezionata dall'utente.

### Componente di visualizzazione dei dati di controllo qualità

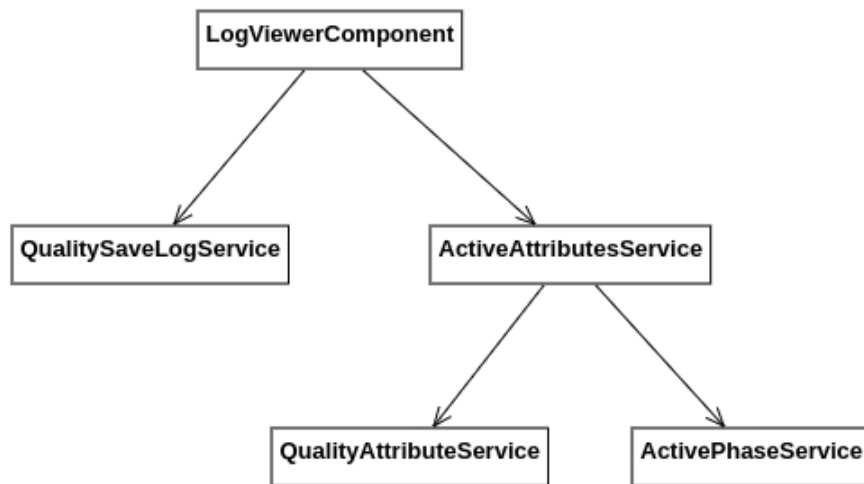


Figura 3.16: Diagramma delle classi del componente di visualizzazione dei dati di qualità

## 3.5 Codifica

Le attività di **codifica** hanno come scopo l'implementazione concreta del *software* in base alla specifica fornita dalle attività di **progettazione**.

Tra le difficoltà incontrate in queste attività, segnalo l'inversione della tabella per la visualizzazione tramite *smartphone* (sezione §3.4.2): l'uso del *framework* di componenti grafici *Angular Material* consente di non doversi occupare dello sviluppo "da zero" di *widget* grafici complessi al prezzo però di una ridotta possibilità di personalizzare i componenti predefiniti; la soluzione adottata sfrutta la possibilità di *scroll* orizzontale del solo contenuto della tabella, "rompendo" parzialmente una delle regole di base dello sviluppo di interfacce grafiche <sup>37</sup>.

Di seguito si riporta il contenuto dei file *TypeScript* relativi alle classi derivate dai casi d'uso descritti nella sezione §3.3.1.

<sup>37</sup><https://mailchimp.com>

### Visualizzazione delle fasi

```
1 this.qualityPhaseService.fetch_2(params)
2   .subscribe({
3     next: (response) => {
4       (response.data != undefined && response.data != null
5         && response.data.length != 0) ? this.phases =
6         response.data : this.openSnackBar(this.
7         translateService.instant("Errore: non ci sono fasi
8           da visualizzare!"), "X");
9     },
10    error: (error) => {
11      const errorDescription = (error.error as ErrorModel)
12        != null ? (error.error as ErrorModel).description
13        : (error.status == 401 ? "Non autorizzato" : "
14          Errore lato server");
15      this.openSnackBar(this.translateService.instant("
16        Errore " + error.status + " - " + errorDescription
17        ), "X");
18      if (error.status == 401) {
19        this.logoutService.logout();
20      }
21      this.loading = false;
22    },
23    complete: () => { this.loading = false; }
24  });
```

**Frammento 3.1:** Visualizzazione delle fasi

### Visualizzazione e valorizzazione degli attributi

```
1 public add(): void {
2     const token: string = this.authInfoService.Token;
3     const qualityvalue: string = this.buildQualityValue();
4     const params = this.prepareAddParams(token, this.
        activePhase.c_projectphase_id!, qualityvalue)
5     this.qualitySaveLogService.Add(params).subscribe({
6         next: (log) => {
7             this.openSuccessSnackBar(this.translateService.
                instant("Inserimento avvenuto correttamente!"), "X
                ");
8             this.mainViewCommunicationsService.viewUpdate.next(
                log);
9         },
10        error: (error) => this.openFailSnackBar("Errore " +
            error.status + " - " + error.error.description, "X")
11    })
12 }
```

**Frammento 3.2:** Aggiunta di dati di controllo qualità

```
1 public update(): void {
2     const token: string = this.authInfoService.Token;
3     const qualityValue = this.buildQualityValue();
4     const params = this.prepareUpdateParams(token, this.
        logToUpdate.c_projectphase_quality_log_id!,
        qualityValue);
5     this.qualitySaveLogService.Update(params).subscribe({
6         next: (log) => {
7             this.openSuccessSnackBar(this.translateService.
                instant("Aggiornamento avvenuto correttamente!"),
                "X")
8             this.mainViewCommunicationsService.viewUpdate.next(
                log);
9         },
10        error: (error) => this.openFailSnackBar("Errore " +
            error.status + " - " + error.error.description, "X")
11    });
12 }
```

**Frammento 3.3:** Modifica di dati di controllo qualità

## Visualizzazione dei dati di qualità

```

1 ngOnInit(): void {
2   this.activeAttributesService.getActiveAttributes()
3     .subscribe(attributes => {
4     this.displayedColumns = attributes.map((attribute) =>
5       attribute.attributevalue!);
6     this.attributes = attributes.slice();
7     this.attributes.push({ attributename: 'Azioni',
8       attributevalue: 'Actions' });
9     if (this.displayedColumns.length == 0) {
10      this.openFailSnackBar("Errore: non sono disponibili
11        attributi per la fase selezionata!", "X");
12    }
13    else if (this.displayedColumns.findIndex(value =>
14      value == 'Actions') == -1) {
15      this.displayedColumns.push("Actions");
16    }
17  });
18  this.mainViewCommunicationsService.viewUpdate.subscribe((
19    updatedLog) => {
20    this.highlighted = {};
21    this.updateTable(this.lastPhase);
22    this.blinkLogId = updatedLog.
23      c_projectphase_quality_log_id!;
24    setTimeout(() => {
25      this.blinkLogId = 0;
26    }, 5500);
27  });
28 }

```

**Frammento 3.4:** Aggiornamento degli attributi caratterizzanti il controllo qualità

```

1 this.qualitySaveLogService.fetch_1(params).subscribe({
2   next: (response) => {
3     let logs: any[] = [];
4     this.activeLogs = response.data!;
5     response.data?.forEach((log) => {
6       const actualQualityValue: { type: string; value:
7         string; } = log.qualityvalue! as any;
8       let aux = JSON.parse(actualQualityValue.value);
9       aux.Actions = "";
10      aux.c_projectphase_quality_log_id = log.
11        c_projectphase_quality_log_id;
12      logs.push(aux);
13    });
14    logs.sort((firstLog, secondLog) => { return secondLog.
15      c_projectphase_quality_log_id - firstLog.
16      c_projectphase_quality_log_id });
17    this.logs.next(logs);
18  }
19 });

```

**Frammento 3.5:** Aggiornamento della tabella di visualizzazione dei dati di controllo qualità

## 3.6 Verifica

Le attività di **verifica** hanno come scopo l'accertamento che:

- Il prodotto *software* in uscita dal processo di **codifica** soddisfi le specifiche stabilite nelle attività di **progettazione**;
- L'esecuzione delle attività di **analisi**, **progettazione** e **codifica** (per un determinato periodo di tempo) non abbia introdotto errori.

La **verifica** si svolge mediante l'uso di due approcci tra loro complementari:

- **Analisi statica**: è un metodo di **verifica** che non prevede l'esecuzione del codice prodotto, si applica sia al codice sorgente sia alla documentazione prodotta e punta all'individuazione di errori nel prodotto quanto prima possibile rispetto all'inizio delle attività di sviluppo;
- **Analisi dinamica**: è un metodo di **verifica** del corretto funzionamento del *software* durante la sua esecuzione, identificando errori, problemi di prestazioni o comportamenti indesiderati attraverso l'esecuzione di *test* specifici.

### Analisi statica

Durante il periodo di *stage*, ho eseguito attività di **analisi statica** manualmente (senza quindi avvalermi di strumenti *software*) per quanto riguarda la documentazione di progetto, data l'assenza di direttive in relazione allo stile di scrittura del codice sorgente: gli obiettivi di tali attività erano l'individuazione e la conseguente correzione di errori ortografici.

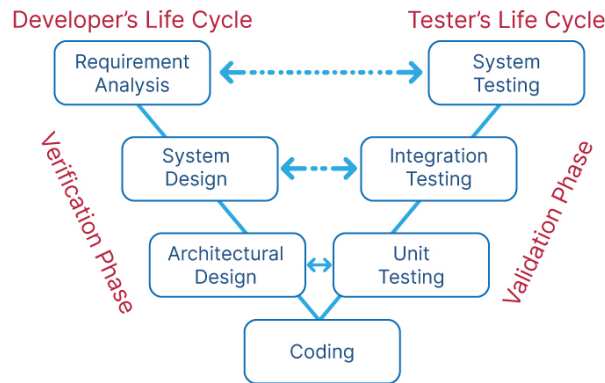
Al termine della scrittura (o modifica) di una porzione di documento unitaria (una sezione o sottosezione di massimo due pagine), seguivano la rilettura e la correzione degli eventuali errori.

### Analisi dinamica

Durante le attività di tirocinio, ho eseguito prove (*test*) di **analisi dinamica** di tre tipologie distinte, seguendo di fatto le indicazioni del modello di sviluppo *software* "a V":

- **Test di unità**: un *test* di unità si può identificare nel compito di controllare se l'esecuzione della più piccola parte utilmente verificabile del codice sorgente (unità) procede come previsto dalla specifica di **progettazione**;
- **Test di integrazione**: un *test* di integrazione è il compito di verificare che l'interazione e combinazione delle diverse unità di codice avvenga come previsto dalle attività di **progettazione**, dopo che esse state messe alla prova singolarmente mediante *test* di unità;
- **Test di sistema**: un *test* di sistema è il controllo delle funzionalità del codice sorgente nel suo complesso, accertando che soddisfi tutti i requisiti stabiliti nelle attività di **analisi**.



Figura 3.17: Modello di sviluppo adottato<sup>38</sup>

Ho sviluppato del codice che concretizzasse in forma di *software* i *test* di unità e di integrazione tramite il *framework Jasmine*: questo rende possibile la loro esecuzione, se l'ambiente di *test* è ben configurato, in maniera ripetibile, automatizzata e rapida. Ho deciso di eseguire i *test* di sistema manualmente data la poca esperienza con il *framework Jasmine* ed il poco tempo rimasto per il completamento delle attività di progetto: ritengo valida questa scelta date le dimensioni ridotte del prodotto e la possibilità di automatizzare queste prove, data la presenza della documentazione necessaria.

### 3.7 Validazione

Le attività di **validazione** hanno come scopo l'accertamento che il prodotto finito sia conforme alle aspettative di chi ha la visione dei bisogni da soddisfare con il prodotto (proponente).

Viene effettuato un collaudo in presenza del committente, ovvero colui (persona fisica, azienda o ente) che ha interesse diretto nel risultato finale del progetto: nel mio caso, le figure del proponente e del committente sono rappresentate dal tutor aziendale *Michele Rigo* ed il collaudo ha avuto luogo durante l'ultima settimana di attività.

In seguito all'accettazione del prodotto, ho potuto dedicare i restanti giorni all'esecuzione di attività di integrazione e studio della libreria di componenti grafici *DevExtreme*, come riportato nella sezione §1.4.

### 3.8 Risultato finale

Riporto alcune statistiche di progetto, seguite da schermate dell'applicazione *ADeQA* e spiegazione, ove necessario, delle differenze tra l'interfaccia grafica realizzata e l'interfaccia grafica progettata.

<sup>38</sup>Fonte: <https://www.shiksha.com>

### 3.8.1 Statistiche qualitative e quantitative

#### Requisiti soddisfatti

Il prodotto sviluppato ricopre tutti i requisiti indicati nelle tabelle della sezione §3.3.2 tranne *RF-3*, *RF-3.1* e *RF-3.2*: questo è dovuto alla mancata esposizione dei servizi necessari per il filtraggio delle fasi di lavorazione.

Tipologia	Quantità	Soddisfatti
Funzionali	27	25
Di qualità	2	2
Prestazionali	2	2
Di vincolo	6	6

Tabella 3.7: Tabella riassuntiva dei requisiti soddisfatti

#### Code coverage

Con "code coverage" si fa riferimento a una metrica utilizzata nel campo dello sviluppo *software* per valutare quanto del codice sorgente di un'applicazione è stato eseguito durante l'esecuzione di un insieme di *test*; non garantisce necessariamente la qualità dei *test* eseguiti e può rilevare la presenza di malfunzionamenti, ma non dimostra la loro assenza (*Dijkstra* <sup>39</sup>).

Le metriche di *code coverage* utilizzate nel progetto sono:

- **Line coverage**: indica la percentuale di linee di codice eseguite durante l'esecuzione dei *test* rispetto al totale delle linee di codice sulle quali la prova viene eseguita;
- **Branch coverage**: percentuale di rami condizionali (decisioni) che sono stati attraversati durante l'esecuzione dei *test* rispetto al totale dei rami presenti nel codice;
- **Statement coverage**: percentuale di istruzioni che sono state eseguite durante l'esecuzione dei *test* rispetto al totale delle istruzioni nel codice;
- **Function coverage**: percentuale di funzioni che sono state "chiamate" durante l'esecuzione dei *test* rispetto al totale delle funzioni nel codice.

```
===== Coverage summary =====
Statements : 79.69% ( 475/596 )
Branches   : 69.03% ( 107/155 )
Functions  : 77.38% ( 130/168 )
Lines      : 79.18% ( 426/538 )
=====
```

Figura 3.18: Code coverage di progetto

<sup>39</sup>Fonte: <https://vitolavecchia.altervista.org>

### Quantità di prodotti

Nelle statistiche sotto riportate, si può notare che la voce "linee di codice" differisce dal totale delle linee di codice testate dell'immagine conclusiva della sezione precedente: il motivo si trova nel fatto che le linee di codice testate non comprendono istruzioni di dichiarazione variabili, importazione di file e, in generale, tutto ciò che non è contenuto all'interno di un metodo.

Metrica	Valore
Documenti prodotti	3
Pagine di documentazione	69
Componenti <i>Angular</i>	11
Servizi <i>Angular</i>	9
<i>Pipes Angular</i>	2
Modelli <i>TypeScript</i>	1
Linee di codice	4582
Numero di file	67

Tabella 3.8: Statistiche quantitative sul prodotto

### 3.8.2 Interfaccia grafica

#### Autenticazione

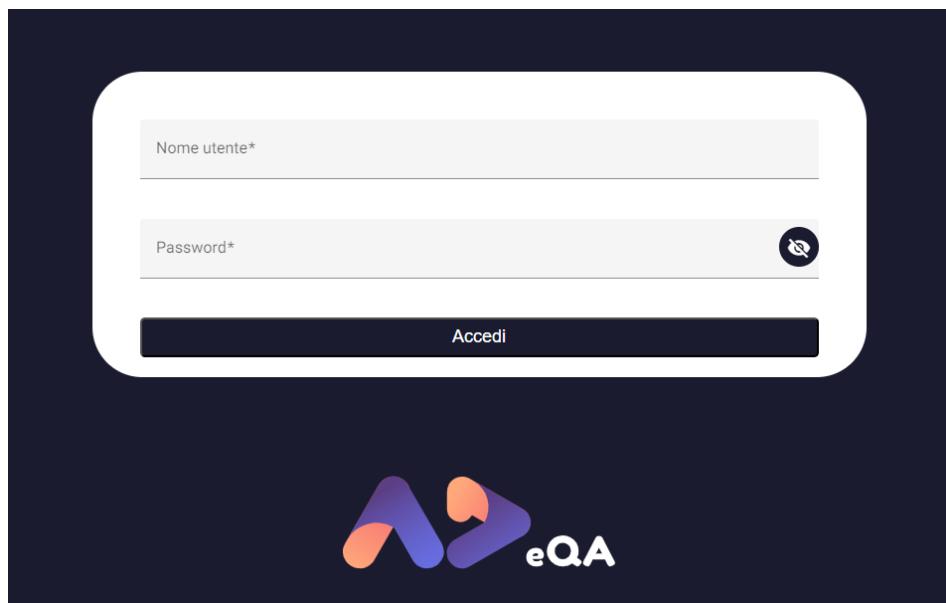


Figura 3.19: Schermata di autenticazione - passo 1

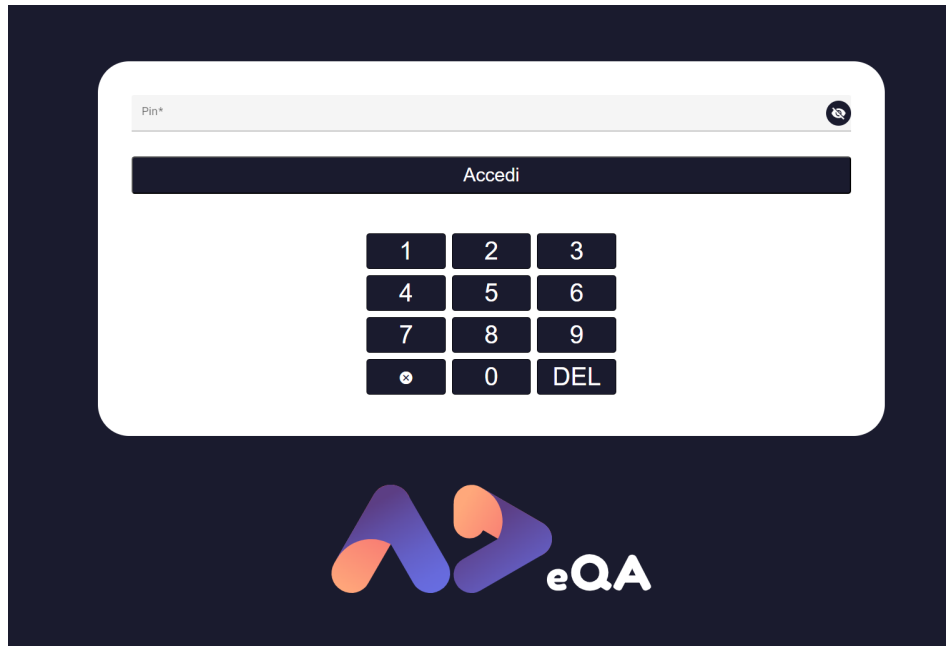


Figura 3.20: Schermata di autenticazione - passo 2

Entrambe le schermate di autenticazione sono state realizzate rispettando le specifiche di progettazione grafica.

### Visualizzazione principale

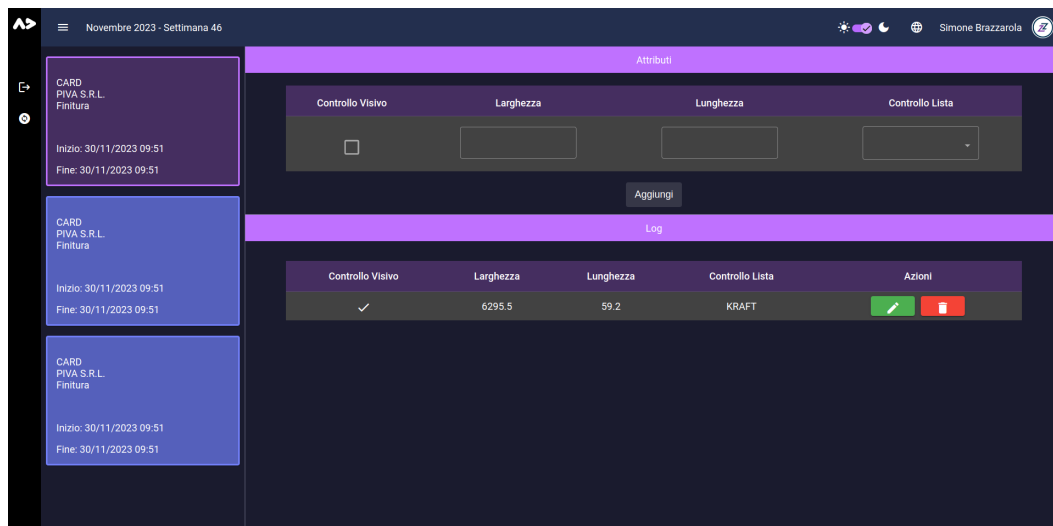


Figura 3.21: Schermata principale

Anche in questo caso, vi è un'importante somiglianza con l'interfaccia ideata nel processo di **progettazione**: dalle figure sottostanti, si possono capire quali sono state le modifiche apportate alle specifiche.

Figura 3.22: Intestazione - visualizzazione *desktop*

Il *widget* di intestazione è stato dotato, a sinistra, di un pulsante grafico che serve per mostrare / far scomparire la lista di fasi di lavorazione.



Per quanto riguarda la visualizzazione *desktop*, la barra laterale contiene un pulsante di disconnessione in più rispetto alla progettazione: questo è stato concesso per evitare la ri-esecuzione del primo passo di autenticazione agli utenti che vogliono autenticarsi come fossero un altro operatore.

Figura 3.23: Barra laterale

Figura 3.24: Intestazione - visualizzazione *mobile*

Per quanto riguarda la visualizzazione per dispositivi mobili, la barra di intestazione (così come la barra laterale per la visualizzazione *desktop*) contiene un pulsante di disconnessione in più rispetto alla progettazione.

## Capitolo 4

# Contesto di svolgimento delle attività

Qui introdurrò brevemente il contenuto delle sezioni sottostanti.

### 4.1 Soddisfacimento degli obiettivi prefissati

In questa sezione metterò in relazione gli obiettivi indicati in §2.5 ed i risultati indicati in §3.8.

### 4.2 Competenze e conoscenze acquisite

In questa sezione descriverò le abilità e le conoscenze acquisite nel corso del tirocinio indicando (se necessario) i benefici ottenuti ed il loro grado di acquisizione.

### 4.3 Competenze curricolari e lavorative

In questa sezione discuterò della differenza tra le competenze acquisite ed erogate dal corso di studi e le competenze necessarie per lo svolgimento delle attività di tirocinio.

## Appendice A

### Metodologie agili

# Acronimi e abbreviazioni

**API** *Application Program Interface*. 47

**DAM** *Digital asset management*. 48

**ERP** *Enterprise resource planning*. 48

**IT** *Information Technology*, acronimo usato per indicare persone o cose attinenti all'ambito informatico. 2

**MES** *Manufacturing execution system*. 48

**PWA** *Progressive Web App*. 14, 49

**REST** *Representational State Transfer*. 49

**UML** *Unified Modeling Language*. 50

**WMS** *Warehouse management system*. 50



# Glossario

**API** in informatica con il termine *Application Programming Interface* (ing. interfaccia di programmazione di un'applicazione) si indicano regole e specifiche per la comunicazione tra software.

Tali regole fungono da interfaccia tra i vari software e ne facilitano l'interazione, allo stesso modo in cui l'interfaccia utente facilita l'interazione tra uomo e computer.

*Application Programming Interface*. URL: [https://www.treccani.it/enciclopedia/api\\_%28Lessico-del-XXI-Secolo%29/](https://www.treccani.it/enciclopedia/api_%28Lessico-del-XXI-Secolo%29/) . 10, 11, 18, 46

**Backend** con il termine "*backend*" si intende la parte non visibile all'utente di un programma, che elabora e gestisce i dati generati dall'interfaccia grafica.

*Backend*. URL: [https://it.wikipedia.org/wiki/Front-end\\_e\\_back-end](https://it.wikipedia.org/wiki/Front-end_e_back-end) . 9, 11

**Black box** con il termine "*black box*" si intende un dispositivo con riferimento alle sole caratteristiche esterne, in particolare alle funzioni di trasferimento tra grandezze di ingresso o di uscita, ignorando cioè del tutto la costituzione interna.

*Black box*. URL: <https://www.treccani.it/vocabolario/scatola/> . 9

**Business to business** con il termine "*business to business*" (it. commercio interaziendale) si intende la tipologia di commercio elettronico che intercorre tra attori economici organizzati in forma d'impresa, quali per esempio le aziende manifatturiere, industriali e commerciali, attraverso siti *web* dedicati.

*Business to business*. URL: [https://www.treccani.it/enciclopedia/b2b\\_%28Lessico-del-XXI-Secolo%29/](https://www.treccani.it/enciclopedia/b2b_%28Lessico-del-XXI-Secolo%29/) . 2

**Controllo qualità** con il termine "controllo qualità" si riferisce alle fasi del sistema di gestione della qualità che prevedono ispezioni, test, esami e verifiche mirate a determinare il livello di soddisfacimento dei requisiti stabiliti per un determinato prodotto, servizio o processo.

*Controllo qualità*. URL: <https://www.teknoring.com/wikitecnica/tecnologia/controllo-di-qualita/> . iii

**Digital asset** con il termine "*digital asset*" (it. risorsa digitale) si intende tutto ciò che esiste solo in forma digitale e viene fornito con un diritto di utilizzo distinto o un'autorizzazione in base all'uso.

I dati che non possiedono tale diritto non sono considerati beni. *Digital asset*. URL: [https://en.wikipedia.org/wiki/Digital\\_asset](https://en.wikipedia.org/wiki/Digital_asset) . 1, 3

**Digital asset management** con il termine "*digital asset management*" (it. sistema di gestione delle risorse digitali) si intende un *software* che consente di creare, organizzare e distribuire i contenuti su differenti canali e aumentare l'efficacia della comunicazione.

È utilizzato per centralizzare e organizzare le risorse in un'unica libreria di facile accesso. *Digital asset management*. URL: <https://onpage.it/differenza-tra-pim-e-dam-per-aziende/> . 3, 46

**Enterprise resource planning** con il termine "*enterprise resource planning*" (it. pianificazione delle risorse d'impresa) si intende un *software* di gestione che integra tutti i processi aziendali e tutte le funzioni aziendali rilevanti, ad esempio vendite, acquisti, gestione magazzino, finanza o contabilità.

*Enterprise resource planning*. URL: [https://it.wikipedia.org/wiki/Enterprise\\_resource\\_planning](https://it.wikipedia.org/wiki/Enterprise_resource_planning) . 2, 46

**Filiera produttiva** con il termine "filiera produttiva" si indica la sequenza delle lavorazioni (detta anche filiera tecnologico-produttiva), effettuate in successione, al fine di trasformare le materie prime in un prodotto finito (ingl. *supply chain*). Le diverse imprese sono integrate tra loro (ai fini della realizzazione di un prodotto):

- **Verticalmente:** se svolgono una o più attività della filiera;
- **Orizzontalmente:** se operano allo stesso stadio di un ciclo produttivo.

Con la globalizzazione dell'economia, esse possono essere situate in paesi e continenti diversi.

*Filiera produttiva*. URL: <https://www.treccani.it/enciclopedia/filiera-produttiva/>. iii

**Frontend** con il termine "*frontend*" si intende la parte visibile all'utente di un programma e con cui egli può interagire, tipicamente un'interfaccia utente.

*Frontend*. URL: [https://it.wikipedia.org/wiki/Front-end\\_e\\_back-end](https://it.wikipedia.org/wiki/Front-end_e_back-end) . 11

**Innovazione** con il termine "innovazione" si intende l'atto e l'effetto dell'innovare, cioè dell'introdurre concetti, metodi, strumenti nuovi.

*Innovazione*. URL: [https://www.treccani.it/enciclopedia/innovazione\\_\(Dizionario-delle-Scienze-Fisiche\)/](https://www.treccani.it/enciclopedia/innovazione_(Dizionario-delle-Scienze-Fisiche)/) . 6, 7

**Manufacturing execution system** con il termine "*manufacturing execution system*" (it. sistema di esecuzione manifatturiera) si intende un *software* che ha la principale funzione di gestire e controllare la funzione produttiva di un'azienda. La gestione riguarda il dispaccio degli ordini, gli avanzamenti in quantità e tempo, il versamento a magazzino, nonché il collegamento diretto ai macchinari per dedurre informazioni utili ad integrare l'esecuzione della produzione come a produrre informazioni per il controllo della produzione stessa.

*Manufacturing execution system*. URL: [https://it.wikipedia.org/wiki/Manufacturing\\_Execution\\_System](https://it.wikipedia.org/wiki/Manufacturing_Execution_System) . 3, 46

**Progressive Web App** in informatica con il termine "*Progressive Web App*" (it. applicazione web progressiva) si indica un'applicazione sviluppata utilizzando tecnologie utilizzate solitamente per lo sviluppo web, ma che offre un'esperienza utente simile a quella di un'app nativa:

- Come un sito web, può funzionare su piattaforme e dispositivi diversi utilizzando un unico codice sorgente;
- Come un'app specifica per una piattaforma, può essere installata sul dispositivo, può operare offline e in background, e può integrarsi con il dispositivo e con altre app installate.

*Progressive Web App*. URL: [https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps). iii, 9, 11, 25, 46

**Realtà aumentata** con il termine "realtà aumentata" si intende la tecnica attraverso cui si aggiungono informazioni alla scena reale.

Questa tecnica è realizzabile attraverso piccoli visori sostenuti, come i caschi immersivi, da supporti montati sulla testa che permettono di vedere la scena reale attraverso lo schermo semitrasparente del visore (*see-through*), utilizzato anche per mostrare grafica e testi generati dal computer.

*Realtà aumentata*. URL: [https://www.treccani.it/enciclopedia/realta-aumentata\\_%28Lessico-del-XXI-Secolo%29/](https://www.treccani.it/enciclopedia/realta-aumentata_%28Lessico-del-XXI-Secolo%29/) . 6

**REST** in informatica con il termine *Representational State Transfer* si intende uno stile architetturale di comunicazione tra *software* dotato dei seguenti principi:

1. **Assenza di stato:** ogni richiesta del *client* al *server* deve contenere tutte le informazioni necessarie per comprendere e elaborare la richiesta. Lo stato del *client* non è memorizzato sul *server* tra le richieste;
2. **Architettura *client-server*:** il sistema è diviso in due parti indipendenti: il *client*, che si occupa dell'interfaccia utente e delle interazioni dell'utente, e il *server*, che gestisce la logica aziendale e conserva le risorse;
3. **Possibilità di salvare in *cache* le risposte:** le risposte del *server* devono essere esplicitamente contrassegnate come *cacheable* o *non cacheable*. Ciò consente ai *client* di memorizzare in modo efficiente le risorse e migliorare le prestazioni complessive del sistema;
4. **Interfaccia uniforme:** l'interfaccia tra il *client* e il *server* per l'ottenimento della stessa risorsa deve essere uniforme, non importa da dove proviene la richiesta;
5. **Sistema stratificato:** l'architettura può essere suddivisa in livelli, con ogni livello che svolge un ruolo specifico.

*Representational State Transfer*. URL: <https://www.ibm.com/topics/rest-apis> . 10, 46

**Software house** con il termine "*software house*" si intende un'azienda specializzata nella produzione di software il cui obiettivo è quello di sviluppare applicazioni

informatiche personalizzate per i propri clienti, che possano soddisfare le loro esigenze specifiche; si occupa dell'intero processo di sviluppo: dalle attività di analisi e progettazione, alla scrittura del codice, alla messa in produzione e manutenzione.

*Software house.* URL: <https://www.businesscompetence.it/cose-una-software-house/>. 1

**Stack tecnologico** con il termine "*stack tecnologico*" (it. pila di tecnologie) si intende l'insieme delle tecnologie utilizzate durante lo sviluppo, la manutenzione, il rilascio di un prodotto software; queste tecnologie possono essere i linguaggi di programmazione, i *frameworks*, le librerie e, in generale, gli strumenti utilizzati nei processi citati.

*Stack tecnologico.* URL: <https://www.heap.io/topics/what-is-a-tech-stack>. 2, 8

**Stakeholder** con il termine "*stakeholder*" (it. portatore di interessi) si intendono tutti i soggetti, individui od organizzazioni, attivamente coinvolti in un'iniziativa economica (progetto, azienda), il cui interesse è negativamente o positivamente influenzato dal risultato dell'esecuzione, o dall'andamento, dell'iniziativa e la cui azione o reazione a sua volta influenza le fasi o il completamento di un progetto o il destino di un'organizzazione.

*Stakeholder.* URL: <https://www.treccani.it/enciclopedia/stakeholder/>. 4, 14

**Standalone** con il termine "*standalone*" si intende un oggetto o un *software* capace di funzionare da solo o in maniera indipendente da altri oggetti o *software*, con cui potrebbe altrimenti interagire.

*Standalone.* URL: [https://it.wikipedia.org/wiki/Stand-alone\\_\(informatica\)](https://it.wikipedia.org/wiki/Stand-alone_(informatica)). 10

**UML** in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 18, 46

**Warehouse management system** con il termine "*warehouse management system*" (it. sistema di gestione del magazzino) si intende un *software* che aiuta le aziende a gestire e controllare le operazioni quotidiane di magazzino, dall'ingresso delle merci e materiali in un centro di distribuzione o polo logistico fino alla loro uscita.

*Warehouse management system.* URL: <https://www.sap.com/italy/products/scm/extended-warehouse-management/what-is-a-wms.html>. 2, 46

# Bibliografia

## Siti web consultati

- Application Programming Interface*. URL: [https://www.treccani.it/enciclopedia/api\\_%28Lessico-del-XXI-Secolo%29/](https://www.treccani.it/enciclopedia/api_%28Lessico-del-XXI-Secolo%29/) (cit. a p. 47).
- Backend*. URL: [https://it.wikipedia.org/wiki/Front-end\\_e\\_back-end](https://it.wikipedia.org/wiki/Front-end_e_back-end) (cit. a p. 47).
- Black box*. URL: <https://www.treccani.it/vocabolario/scatola/> (cit. a p. 47).
- Business to business*. URL: [https://www.treccani.it/enciclopedia/b2b\\_%28Lessico-del-XXI-Secolo%29/](https://www.treccani.it/enciclopedia/b2b_%28Lessico-del-XXI-Secolo%29/) (cit. a p. 47).
- Controllo qualità*. URL: <https://www.teknoring.com/wikitecnica/tecnologia/controllo-di-qualita/> (cit. a p. 47).
- Digital asset*. URL: [https://en.wikipedia.org/wiki/Digital\\_asset](https://en.wikipedia.org/wiki/Digital_asset) (cit. a p. 47).
- Digital asset management*. URL: <https://onpage.it/differenza-tra-pim-e-dam-per-aziende/> (cit. a p. 48).
- Enterprise resource planning*. URL: [https://it.wikipedia.org/wiki/Enterprise\\_resource\\_planning](https://it.wikipedia.org/wiki/Enterprise_resource_planning) (cit. a p. 48).
- Filiera produttiva*. URL: <https://www.treccani.it/enciclopedia/filiera-produttiva/> (cit. a p. 48).
- Frontend*. URL: [https://it.wikipedia.org/wiki/Front-end\\_e\\_back-end](https://it.wikipedia.org/wiki/Front-end_e_back-end) (cit. a p. 48).
- Innovazione*. URL: [https://www.treccani.it/enciclopedia/innovazione\\_\(Dizionario-delle-Scienze-Fisiche\)/](https://www.treccani.it/enciclopedia/innovazione_(Dizionario-delle-Scienze-Fisiche)/) (cit. a p. 48).
- Manufacturing execution system*. URL: [https://it.wikipedia.org/wiki/Manufacturing\\_Execution\\_System](https://it.wikipedia.org/wiki/Manufacturing_Execution_System) (cit. a p. 48).
- Progressive Web App*. URL: [https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps) (cit. a p. 49).
- Realtà aumentata*. URL: [https://www.treccani.it/enciclopedia/realta-aumentata\\_%28Lessico-del-XXI-Secolo%29/](https://www.treccani.it/enciclopedia/realta-aumentata_%28Lessico-del-XXI-Secolo%29/) (cit. a p. 49).
- Representational State Transfer*. URL: <https://www.ibm.com/topics/rest-apis> (cit. a p. 49).

*Software house*. URL: <https://www.businesscompetence.it/cose-una-software-house/> (cit. a p. 50).

*Stack tecnologico*. URL: <https://www.heap.io/topics/what-is-a-tech-stack> (cit. a p. 50).

*Stakeholder*. URL: <https://www.treccani.it/enciclopedia/stakeholder/> (cit. a p. 50).

*Standalone*. URL: [https://it.wikipedia.org/wiki/Stand-alone\\_\(informatica\)](https://it.wikipedia.org/wiki/Stand-alone_(informatica)) (cit. a p. 50).

*Warehouse management system*. URL: <https://www.sap.com/italy/products/scm/extended-warehouse-management/what-is-a-wms.html> (cit. a p. 50).